Journal Subline

LNCS 5390

Transactions on **Rough Sets IX**

James F. Peters · Andrzej Skowron

Editors-in-Chief

Springer

James F. Peters   Andrzej Skowron
Henryk Rybiński (Eds.)

Transactions on Rough Sets IX

Springer

Volume Editors

James F. Peters
University of Manitoba
Department of Electrical and Computer Engineering
Winnipeg, Manitoba, R3T 5V6, Canada
E-mail: jfpeters@ee.umanitoba.ca

Andrzej Skowron
Warsaw University
Institute of Mathematics
Banacha 2, 02-097, Warsaw, Poland
E-mail: skowron@mimuw.edu.pl

Henryk Rybiński
Warsaw University of Technology
Institute of Computer Science
Nowowiejska 15/19, 00-665 Warsaw, Poland
E-mail: H.Rybinski@ii.pw.edu.pl

Library of Congress Control Number: 2008942076

CR Subject Classification (1998): F.4.3, I.5, H.2.8, I.2, G.2

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN	0302-9743 (Lecture Notes in Computer Science)
ISSN	1861-2059 (Transaction on Rough Sets)
ISBN-10	3-540-89875-1 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-89875-7 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12586622 06/3180 5 4 3 2 1 0

Preface

Volume IX of the Transactions on Rough Sets (TRS) provides evidence of the continuing growth of a number of research streams that were either directly or indirectly begun by the seminal work on rough sets by Zdzisław Pawlak (1926-2006)[1]. One of these research streams inspired by Prof. Pawlak is rough set-based intelligent systems, a topic that was an important part of his early 1970s work on knowledge description systems prior to his discovery of rough sets during the early 1980s. Evidence of intelligent systems as a recurring motif over the past two decades can be found in the rough-set literature that now includes over 4,000 publications by more than 1,600 authors in the rough set database[2].

This volume of the TRS includes articles that are extensions of papers included in the first conference on Rough Sets and Intelligent Systems Paradigms[3]. In addition to research on intelligent systems, this volume also presents papers that reflect the profound influence of a number of other research initiatives by Zdzisław Pawlak.

In particular, this volume introduces a number of new advances in the foundations and applications of artificial intelligence, engineering, image processing, logic, mathematics, medicine, music, and science. These advances have significant implications in a number of research areas such as attribute reduction, approximation schemes, category-based inductive reasoning, classifiers, classifying mappings, context algebras, data mining, decision attributes, decision rules, decision support, diagnostic feature analysis, EEG classification, feature analysis, granular computing, hierarchical classifiers, indiscernibility relations, information granulation, information systems, musical rhythm retrieval, probabilistic dependencies, reducts, rough-fuzzy C-means, rough inclusion functions, roughness, singing voice recognition, and vagueness. A total of 47 researchers are represented in this volume.

This volume has been made possible thanks to the laudable efforts of a great many generous persons and organizations. The editors and authors of this volume also extend an expression of gratitude to Alfred Hofmann, Ursula Barth, Christine Günther, and the LNCS staff at Springer for their support in making this volume of the TRS possible. In addition, the editors of this volume extend their thanks to Marcin Szczuka for his consummate skill and care in the compilation of this volume.

[1] See, *e.g.*, Pawlak, Z., Skowron, A.: Rudiments of rough sets, *Information Sciences* 177 (2007) 3-27; Pawlak, Z., Skowron, A.: rough sets: Some extensions, *Information Sciences* 177 (2007) 28-40; Pawlak, Z., Skowron, A.: Rough sets and Boolean reasoning, *Information Sciences* 177 (2007) 41-73.

[2] `http://rsds.wsiz.rzeszow.pl/rsds.php`

[3] Int. Conf. on Rough Sets and Emerging Intelligent Systems Paradigms, *Lecture Notes in Artificial Intelligence* 4585. Springer, Berlin, 2007.

The editors of this volume were supported by the by the Ministry of Science and Higher Education of the Republic of Poland, research grants No. NN516 368334 and 3T11C 002 29, by the by Ministry of Regional Development of the Republic of Poland, grant "Decision Support - New Generation Systems" of Innovative Economy Operational Programme 2007-2013 (Priority Axis 1. Research and development of new technologies), and the Natural Sciences and Engineering Research Council of Canada (NSERC) research grant 185986.

October 2008

Henryk Rybiński
James F. Peters
Andrzej Skowron

LNCS Transactions on Rough Sets

This journal subline has as its principal aim the fostering of professional exchanges between scientists and practitioners who are interested in the foundations and applications of rough sets. Topics include foundations and applications of rough sets as well as foundations and applications of hybrid methods combining rough sets with other approaches important for the development of intelligent systems.

The journal includes high-quality research articles accepted for publication on the basis of thorough peer reviews. Dissertations and monographs up to 250 pages that include new research results can also be considered as regular papers. Extended and revised versions of selected papers from conferences can also be included in regular or special issues of the journal.

Table of Contents

Vagueness and Roughness

Zbigniew Bonikowski[1] and Urszula Wybraniec-Skardowska[2]

[1] Institute of Mathematics and Informatics
University of Opole, Opole, Poland
zbonik@math.uni.opole.pl
[2] Autonomous Section of Applied Logic,
Poznań School of Banking, Faculty in Chorzów, Poland
uws@uni.opole.pl

Abstract. The paper proposes a new formal approach to vagueness and vague sets taking inspirations from Pawlak's rough set theory. Following a brief introduction to the problem of vagueness, an approach to conceptualization and representation of vague knowledge is presented from a number of different perspectives: those of logic, set theory, algebra, and computer science. The central notion of the vague set, in relation to the rough set, is defined as a family of sets approximated by the so called lower and upper limits. The family is simultaneously considered as a family of all denotations of sharp terms representing a suitable vague term, from the agent's point of view. Some algebraic operations on vague sets and their properties are defined. Some important conditions concerning the membership relation for vague sets, in connection to Blizard's multisets and Zadeh's fuzzy sets, are established as well. A classical outlook on a logic of vague sentences (vague logic) based on vague sets is also discussed.

Keywords: vagueness, roughness, vague sets, rough sets, knowledge, vague knowledge, membership relation, vague logic.

1 Introduction

Logicians and philosophers have been interested in the problem area of vague knowledge for a long time, looking for some logical foundations of a theory of vague notions (terms) constituting such knowledge. Recently vagueness and, more generally - imperfection, has become the subject of investigations of computer scientists interested in the problems of AI, in particular, in the problems of reasoning on the basis of imperfect information and in the application of computers to support and represent such reasoning in the computer memory (see, *e.g.*, Parsons [15]).

Imperfection is considered in a general information-based framework, where objects are described by an agent in terms of attributes and their values. Bonissone and Tong [5] indicated three types of imperfections relating to information: *incompleteness*, *uncertainty* and *imprecision*. Incompleteness arises from the absence of a value of an attribute for some objects. Uncertainty arises from a lack of information; as a result, an object's attribute may have a finite set of values rather than a single value. Imprecision occurs when an attribute's value cannot be measured with adequate precision. There are also other classifications of imperfect information (see, *e.g.*, Słowiński, Stefanowski [26]).

Marcus [12] thought of imprecision more generally. He distinguished, *e.g.*, such types of imprecision as *vagueness*, *fuzziness* and *roughness*. Both fuzziness and roughness are mathematical models of vagueness.

Fuzziness is closely related to Zadeh's fuzzy sets [28]. In fuzzy set theory, vagueness is described by means of a specific membership relation. Fuzziness is often identified with vagueness, however, Zadeh [29] noted that vagueness comprises fuzziness. Roughness is connected with Pawlak's rough sets [19].

Classical, set-theoretical sets (orthodox sets) are not sufficient to deal with vagueness. Non-orthodox sets - rough sets and fuzzy sets - are used in two different approaches to vagueness (Pawlak [22]): while Zadeh's fuzzy set theory represents a quantitative approach, Pawlak's rough set theory represents a qualitative approach to vagueness.

Significant results obtained by computer scientists in the area of imprecision and vagueness, such as Zadeh's fuzzy set theory [28], Shafer's theory of evidence [24] and Pawlak's rough set theory [19,21], greatly contributed to advancing and intensifying of research into vagueness.

This paper is an extended version of a previous article by the same authors [4]. It proposes a new approach to vagueness taking into account the main ideas of roughness. Roughness considered as a mathematical model of vagueness is here replaced by an approach to vagueness in which vague sets, defined in this paper, play the role of rough sets. Vague sets are connected with vague knowledge and, at the same time, are understood as denotations of vague notions. The paper also attempts to lay logical foundations to the theory of vague notions (terms) and thus bring an essential contribution to research in this area.

The structure of the paper is as follows. In Sect. 2, we introduce the notion of unit information (unit knowledge) and vague information (vague knowledge). The central notion of the vague set, inspired by Pawlak's notion of a rough set, is defined in Sect. 3. Section 4 is devoted to the problem of multiplicity of an object's membership to a vague set. In Sect. 5 some operations on vague sets and their algebraic properties are given. A view on the logic of vague concepts (terms) is discussed in Sect. 6. The paper ends with Sect. 7 which delivers some final remarks.

2 Unit Knowledge and Vague Knowledge

In the process of cognition of a definite fragment of reality, the cognitive agent (a man, an expert, a group of men or experts, a robot) attempts to discover information contained in it or, more adequately, about its objects. Each fragment of reality recognized by the agent can be interpreted as the following relational structure:

$$\Re = \langle \mathcal{U}, \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n \rangle, \tag{1}$$

where $\mathcal{U}$, the *universe of objects of reality* $\Re$, is a non-empty set, and $\mathcal{R}_i$, for $i = 1, 2, \dots, n$, is the set of i-ary relations on $\mathcal{U}$. One-ary relations are regarded as subsets of $\mathcal{U}$ and understood as properties of objects of $\mathcal{U}$, and multi-argument relations as relationships among its objects. Formally, every k-ary relation of $\mathcal{R}_k$ is a subset of $\mathcal{U}^k$.

We assume that reality $\Re$ is objective with respect to cognition. Objective knowledge about it consists of pieces of unit information (knowledge) about objects of $\mathcal{U}$ with respect to all particular relations of $\mathcal{R}_k$ $(k = 1, 2, \dots, n)$.

We introduce the notion of knowledge and vague knowledge in accordance with some conceptions of the second co-author of this paper ([27]).

Definition 1. Unit information (knowledge).
Unit information (knowledge) about the object $o \in \mathcal{U}$ with respect to the relation $R \in \mathcal{R}_k$ $(k = 1, 2, \dots, n)$ is the image $\overrightarrow{R}(o)$ of the object o with respect to the relation R[1].

Discovering unit knowledge about objects of reality $\Re$ is realized through asking questions which include certain aspects, called ***attributes***, of the objects of the universe $\mathcal{U}$. Then, we usually choose a finite set $U \subseteq \mathcal{U}$ as the universe and we put it forward as a generalized *attribute-value system* Σ, also called an *information system* (cf. Codd [6]; Pawlak [16], [18], [19]; Marek and Pawlak [13]). Its definition is as follows:

Definition 2. Information system.
Σ is an *information system* iff it is an ordered system

$$\Sigma = \langle U, A_1, A_2, \dots, A_n \rangle, \tag{2}$$

where $U \subseteq \mathcal{U}$, $\mathrm{card}(U) < \omega$ and A_k $(k = 1, 2, \dots, n)$ is the set of k-ary attributes understood as k-ary functions, i.e.

$$\forall_{a \in A_k} a \colon U^k \to V_a, \tag{3}$$

where V_a is the set of all values of the attribute a.

Example 1. Let us consider the following information system: $\mathbf{S} = \langle S, A_1, A_2 \rangle$, where $S = \{p_1, p_2, \dots, p_5\}$ is a set of 5 papers and $A_1 = \{$IMPACT FACTOR (IF), QUOTATIONS $(Q)\}$, $A_2 = \{$TOPIC CONNECTION $(TC)\}$. The attribute IF is a function which assigns to every paper $p \in S$ an impact factor of the journal in which p was published. We assume that $V_{IF} = [0, 100]$. The value of the attribute Q for any paper $p \in S$ is the number of quotations of p. We assume that $V_Q = \{0, 1, 2, \dots, 2000\}$. We also assume that TC assigns to every pair of papers a quotient of the number of common references by the number of all references, and that $V_{TC} = [0, 1]$.

---

[1] $\overrightarrow{R}(o) = \begin{cases} R, \text{ if } o \in R, \\ \emptyset, \text{ otherwise.} \end{cases}$ for $R \in \mathcal{R}_1$.

$\overrightarrow{R}(o) = \{\langle x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_k \rangle : \langle x_1, \dots, x_{i-1}, o, x_{i+1}, \dots, x_k \rangle \in R\}$ for $R \in \mathcal{R}_k$ $(k = 2, \dots, n)$.

The information system **S** can be clearly presented in the following tables:

	IF	Q
p_1	0.203	125
p_2	0.745	245
p_3	0.498	200
p_4	0.105	150
p_5	1.203	245

TC	p_1	p_2	p_3	p_4	p_5
p_1	1	3/10	0	6/7	0
p_2	3/10	1	0	0	4/17
p_3	0	0	1	0	1/12
p_4	6/7	0	0	1	0
p_5	0	4/17	1/12	0	1

Every attribute of the information system Σ and every value of this attribute explicitly indicates a relation belonging to the so-called ***relational system determined by*** Σ. The unit information (knowledge) about an object $o \in U$ should be considered with respect to relations of the system.

Definition 3. System determined by the information system.
$\Re(\Sigma)$ is a *system determined by the information system* Σ (see (2)) iff

$$\Re(\Sigma) = \langle U, \{R_{a,W} : a \in A_1, \emptyset \neq W \subseteq V_a\}, \ldots, \{R_{a,W} : a \in A_n, \emptyset \neq W \subseteq V_a\}\rangle,$$

where $R_{a,W} = \{(o_1, o_2, \ldots, o_k) \in U^k : a((o_1, o_2, \ldots, o_k)) \in W\}$
for any $k \in \{1, 2, \ldots, n\}$, $a \in A_k$.

Let us see that $\bigcup\{R_{a,\{v\}} : a \in A_1, v \in V_a\} = U$, i.e. the family $\{R_{a,\{v\}} : a \in A_1, v \in V_a\}$ is a covering of U.

It is easy to see that

Fact 1. *The system Σ uniquely determines the system $\Re(\Sigma)$.*

Example 2. Let **S** be the above given information system. Then the system determined by the system **S** is $\Re(\mathbf{S}) = \langle U, R_{A_1}, R_{A_2}\rangle$, where $R_{A_1} = \{R_{IF,S}\}_{\emptyset\neq S\subseteq V_{IF}} \cup \{R_{Q,S}\}_{\emptyset\neq S\subseteq V_Q}$ and $R_{A_2} = \{R_{TC,S}\}_{\emptyset\neq S\subseteq V_{TC}}$.

For any attribute a of the system **S** and any $i, j \in \mathbb{R}$ we adopt the following notation:

$S_i^j = \{v \in V_a : i \leq v \leq j\}$, $S^j = \{v \in V_a : v \leq j\}$, $S_i = \{v \in V_a : i \leq v\}$.

Then, in particular, we can easily state that: $R_{IF,S_{0.1}^{0.5}} = \{p_1, p_3, p_4\}$, $R_{IF,S_{0.7}} = \{p_2, p_5\}$, $R_{IF,S^{0.3}} = \{p_1, p_4\}$, $R_{Q,S_{150}^{150}} = R_{Q,\{150\}} = \{p_4\}$, $R_{Q,S_{200}} = \{p_2, p_3, p_5\}$ and $R_{TC,\{1/12\}} = \{(p_3, p_5), (p_5, p_3)\}$, $R_{TC,\{1\}} = \{(p_i, p_i)\}_{i=1,\ldots,5}$.

The notion of knowledge about the attributes of the system Σ depends on the cognitive agent discovering the fragment of reality Σ. According to Skowron's understanding of the notion of knowledge determined by any unary attribute (cf. Pawlak [17], Skowron et al. [25], Demri, Orlowska [8] pp.16–17), we can adopt the following generalized definition of the notion of ***knowledge K_a about any k-ary attribute a***:

Definition 4. Knowledge K_a about the attribute a.

Let Σ be the information system satisfying (2) and $a \in A_k$ $(k = 1, 2, \ldots, n)$. Then

(a) $K_a = \{((o_1, o_2, \ldots, o_k), V_{a,u}) : u = (o_1, o_2, \ldots, o_k) \in U^k\}$,
where $V_{a,u} \subseteq P(V_a)$, $V_{a,u}$ is the family of all sets of possible values of the attribute a for the object u from the viewpoint of the agent and $P(V_a)$ is the family of all subsets of V_a.

(b) The knowledge K_a of the agent about the attribute a and its value for the object u is
  (0) ***empty*** if $\mathrm{card}(\bigcup_{W \in V_{a,u}} W) = 0$,
  (1) ***definite*** if $\mathrm{card}(\bigcup_{W \in V_{a,u}} W) = 1$,
  (> 1) ***imprecise***, in particular ***vague***, if $\mathrm{card}(\bigcup_{W \in V_{a,u}} W) > 1$.

Let us observe that vague knowledge about some attribute of the information system Σ is connected with the assignation of a ***vague value*** to the object u.

Example 3. Let us consider again the information system $\mathbf{S} = \langle S, A_1, A_2 \rangle$. The agent's knowledge K_{IF}, K_Q, K_{TC} about the attributes of the information system $\mathbf{S}$ can be characterized by means of the following tables:

	$V_{IF,p}$	$V_{Q,p}$
p_1	$\{S_{0.2}, S_{0.3}, S_{0.25}\}$	$\{S_{100}, S_{150}, S_{90}, S_{80}\}$
p_2	$\{S_{0.5}, S_{0.7}, S_{0.8}\}$	$\{S_{180}, S_{200}, S_{250}, S_{240}\}$
p_3	$\{S_{0.5}, S_{0.6}, S_{0.4}\}$	$\{S_{170}, S_{230}, S_{180}, S_{150}\}$
p_4	$\{S_{0.1}, S_{0.2}, S_{0.15}\}$	$\{S_{100}, S_{90}, S_{10}, S_{140}\}$
p_5	$\{S_{0.7}, S_{1.5}, S_{1.0}\}$	$\{S_{270}, S_{150}, S_{240}, S_{200}\}$

$V_{TC,(p,p')}$	p_1	p_2	p_3	p_4	p_5
p_1	$\{S_1^1\}$	$\{S^{0.3}, S^{0.5}\}$	$\{S^{0.1}, S^{0.2}\}$	$\{S_{0.5}, S_{0.8}\}$	$\{S^{0.1}, S^{0.2}\}$
p_2	$\{S^{0.3}, S^{0.5}\}$	$\{S_1^1\}$	$\{S^{0.1}, S^{0.2}\}$	$\{S^{0.1}, S^{0.2}\}$	$\{S^{0.3}, S^{0.4}\}$
p_3	$\{S^{0.1}, S^{0.2}\}$	$\{S^{0.1}, S^{0.2}\}$	$\{S_1^1\}$	$\{S^{0.1}, S^{0.2}\}$	$\{S^{0.3}, S^{0.1}\}$
p_4	$\{S_{0.5}, S_{0.8}\}$	$\{S^{0.1}, S^{0.2}\}$	$\{S^{0.1}, S^{0.2}\}$	$\{S_1^1\}$	$\{S^{0.1}, S^{0.2}\}$
p_5	$\{S^{0.1}, S^{0.2}\}$	$\{S^{0.3}, S^{0.4}\}$	$\{S^{0.3}, S^{0.1}\}$	$\{S^{0.1}, S^{0.2}\}$	$\{S_1^1\}$

From Definitions 1 and 3 we arrive at:

Fact 2. *Unit information (knowledge) about the object $o \in U$ with respect to a relation R of the system $\Re(\Sigma)$ is the image $\overrightarrow{R}(o)$ of the object o with respect to the relation R.*

Contrary to the objective unit knowledge $\overrightarrow{R}(o)$ about the object o of U in the reality $\Re$ with regard to its relation R, the subjective unit knowledge (the unit knowledge of an agent) about the object o of U in the reality $\Re(\Sigma)$ depends on an attribute of Σ determining the relation R and its possible values from the viewpoint of the knowledge of the agent discovering $\Re(\Sigma)$. The subjective unit knowledge $\overrightarrow{R_{ag}}(o)$ depends on the agent's ability to solve the following equation:

$$\overrightarrow{R_{ag}}(o) = x, \qquad (e)$$

where x is an unknown quantity.

Solutions of (e) for a k-ary relation R should be images of the object o with respect to k-ary relations $R_{a,W}$ from $\Re(\Sigma)$, where $\emptyset \neq W \subseteq V_a$. Let us note that for each unary relation R solutions of (e) are unary relations $R_{a,W}$, where $\emptyset \neq W \in V_{a,o}$.

A solution of the equation (e) can be correct – then the agent's knowledge about the object o is ***exact***. If the knowledge is ***inexact***, then at least one solution of (e) is not an image of the object o with respect to the relation R.

Definition 5. Empty, definite and imprecise unit knowledge.
Unit knowledge of the agent about the object $o \in U$ in $\Re(\Sigma)$ with respect to its relation R is

(0) *empty* iff the equation (e) does not have a solution for the agent (the agent knows nothing about the value of the function $\overrightarrow{R}$ for the object o),

(1) *definite* iff the equation (e) has exactly one solution for the agent (either the agent's knowledge is exact – the agent knows the value of the function $\overrightarrow{R}$ for the object o – or he accepts only one, but not necessarily accurate, value of the function),

(> 1) *imprecise* iff the equation (e) has at least two solutions for the agent (the agent allows at least two possible values of the function $\overrightarrow{R}$ for the object o).

From Definitions 4 and 5 we arrive at:

Fact 3. *Unit knowledge of the agent about the object $o \in U$ in $\Re(\Sigma)$ with respect to its relation R is*

(0) ***empty*** *if the agent's knowledge K_a about the attribute a and its value for the object o is empty,*

(1) ***definite*** *if the agent's knowledge K_a about the attribute a and its value for the object o is definite,*

(> 1) ***imprecise*** *if the agent's knowledge K_a about the attribute a and its value for the object o is imprecise.*

When the unit knowledge of the agent about the object o is imprecise, then most often we replace the unknown quantity x in (e) with a vague value.

Example 4. Consider the relation $R = R_{Q,S_{200}}$ within the previous system $\Re(\mathbf{S})$, i.e. the set of all papers of S that have been quoted in at least 200 other papers. The unit knowledge about the paper p_5 with respect to R can be the following vague information:

$$\overrightarrow{R_{ag}}(p_5) = VALUABLE, \tag{e_1}$$

where $VALUABLE$ is an unknown, indefinite, vague quantity.

Then the agent refers to the paper p_5 non-uniquely, assigning to him different images of the paper p_5 with respect to the relations that are possible from his point of view. Then the equation (e_1) usually has, for him, at least two solutions. From *Example 3*, it follows that each of these relations: $R_{Q,S_{270}}, R_{Q,S_{150}}, R_{Q,S_{240}}, R_{Q,S_{200}}$ can be a solution to (e_1). Let us observe that $R_{Q,S_{270}} = \emptyset$, $R_{Q,S_{150}} = \{p_2, p_3, p_4, p_5\}$, $R_{Q,S_{240}} = \{p_2, p_5\}$, $R_{Q,S_{200}} = \{p_2, p_3, p_5\}$.

3 Vague Sets and Rough Sets

Let $\Re(\Sigma)$ be the system determined by the information system Σ. In order to simplify our considerations in the subsequent sections of the paper, we will limit ourselves to the unary relation R (property) – a subset of U of the system $\Re(\Sigma)$.

Definition 6. Inexact unit knowledge of the agent.
Unit knowledge of the agent about the object o in $\Re(\Sigma)$ with respect to R is *inexact* iff the equation (e) has for him at least one solution and at least one of the solutions is not an image $\overrightarrow{R}(o)$.

The equation (e) has then the form:

$$\overrightarrow{R_{ag}}(o) = X, \qquad (ine)$$

where X is an unknown quantity from the viewpoint of the agent, and (ine) has for him at least one solution and at least one of the solutions is not an image $\overrightarrow{R}(o)$.

The equation (ine) can be called the *equation of inexact knowledge of the agent.* All solutions of (ine) are unary relations in the system $\Re(\Sigma)$.

Definition 7. Vague unit knowledge of the agent.
Unit knowledge of the agent about the object o in $\Re(\Sigma)$ with respect to R is *vague* iff the equation (e) has at least two different solutions for the agent.

The equation (e) has then the form:

$$\overrightarrow{R_{ag}}(o) = \mathit{VAGUE}, \qquad (ve)$$

where VAGUE is an unknown quantity, and (ve) has at least two different solutions for the agent.

The equation (ve) can be called the *equation of vague knowledge of the agent.*

Fact 4. *Vague unit knowledge is a particular case of inexact unit knowledge.*

Definition 8. Vague (proper vague) set.
The family of all solutions (sets) of (ine), respectively (ve), is called the *vague set for the object o determined by R*, respectively the *proper vague set for the object o determined by R.*

Example 5. The family of all solutions of (e_1) from *Example 4* is a vague set $\mathbf{V}_{p_5}$ for the paper p_5 determined by $R_{Q,S_{200}}$ and $\mathbf{V}_{p_5} = \{R_{Q,S_{270}}, R_{Q,S_{150}}, R_{Q,S_{240}}, R_{Q,S_{200}}\}$.

Vague sets, thus also proper vague sets, determined by a set R are here some generalizations of sets approximated by representations (see Bonikowski [3]). They are non-empty families of unary relations from $\Re(\Sigma)$ (such that at least one of them includes R) and sub-families of the family $P(U)$ of all subsets of the set U, determined by the set R. They have the greatest lower bound (the *lower limit*) and the least upper bound (the *upper limit*) in $P(U)$ with respect to inclusion. We will denote the greatest lower bound of any family $\mathbf{X}$ by $\underline{\mathbf{X}}$. The least upper bound of $\mathbf{X}$ will be denoted by $\overline{\mathbf{X}}$. So, we can note

Fact 5. *For each vague set* $\mathbf{V}$ *determined by the set (property)* R

$$\mathbf{V} \subseteq \{Y \in P(U) : \underline{\mathbf{V}} \subseteq Y \subseteq \overline{\mathbf{V}}\}. \tag{4}$$

The idea of vague sets was conceived upon Pawlak's idea of rough sets [19], who defined them by means of the operations of *lower approximation*: $_*$ and *upper approximation*: *, defined on subsets of U. The lower approximation of a set is defined as a union of indiscernibility classes of a given relation in U^2 which are included in this set, whereas the upper approximation of a set is defined as a union of the indiscernibility classes of the relation which have a non-empty intersection with this set.

Definition 9. Rough set.
A *rough set* determined by a set $R \subseteq U$ is a family $\mathbf{P}$ of all sets satisfying the condition (5):

$$\mathbf{P} = \{Y \in P(U) : Y_* = R_* \wedge Y^* = R^*\}.^2 \tag{5}$$

Let us observe that because $R \subseteq R \in \mathbf{P}$, the family $\mathbf{P}$ is a non-empty family of sets such that at least one of them includes R (cf. Definition 8). By analogy to Fact 5, we have

Fact 6. *For each rough set* $\mathbf{P}$ *determined by the set (property)* R

$$\mathbf{P} \subseteq \{Y \in P(U) : R_* \subseteq Y \subseteq R^*\}. \tag{6}$$

It is obvious that

Fact 7. *If* $\mathbf{V}$ *is a vague set and* $X_* = \underline{\mathbf{V}}$ *and* $X^* = \overline{\mathbf{V}}$ *for any* $X \in \mathbf{V}$, *then* $\mathbf{V}$ *is a subset of a rough set determined by any set of* $\mathbf{V}$.

For every rough set $\mathbf{P}$ determined by R we have: $\underline{\mathbf{P}} = R_*$ and $\overline{\mathbf{P}} = R^*$. We can therefore consider the following generalization of the notion of the rough set:

Definition 10. Generalized rough set.
A non-empty family $\mathbf{G}$ of subsets of U is called a *generalized rough set* determined by a set R iff it satisfies the condition (7):

$$\underline{\mathbf{G}} = R_* \text{ and } \overline{\mathbf{G}} = R^*. \tag{7}$$

It is easily seen that

Fact 8. *Every rough set determined by a set* R *is a generalized rough set determined by* R.

Fact 9. *If* $\mathbf{V}$ *is a vague set and there exists a set* $X \subseteq U$ *such that* $X_* = \underline{\mathbf{V}}$ *and* $X^* = \overline{\mathbf{V}}$, *then* $\mathbf{V}$ *is a generalized rough set determined by the set* X.

[2] Some authors define a rough set as a pair of sets (lower approximation, upper approximation)(cf., *e.g.*, Iwiński [10], Pagliani [14]).

4 Multiplicity of Membership to a Vague Set

For every object $o \in U$ and every vague set $\mathbf{V}_o$, we can count the ***multiplicity of membership of o to this set***.

Definition 11. Multiplicity of membership.
The number i is the *multiplicity of membership of the object o to the vague set* $\mathbf{V}_o$ iff o belongs to i sets of $\mathbf{V}_o$ ($i \in \mathbb{N}$).

The notion of multiplicity of an object's membership to a vague set is closely related to the so-called ***degree of an object's membership to the set***.

Definition 12. Degree of an object's membership.
Let $\mathbf{V}_o$ be a vague set for the object o and $card(\mathbf{V}_o) = n$. The function μ is called a *degree of membership of o to* $\mathbf{V}_o$ iff

$$\mu(o) = \begin{cases} 0, \text{ if the multiplicity of membership of } o \text{ to } \mathbf{V}_o \text{ equals } 0, \\ \frac{k}{n}, \text{ if the multiplicity of membership of } o \text{ to } \mathbf{V}_o \text{ equals } k \ (0 < k < n), \\ 1, \text{ if the multiplicity of membership of } o \text{ to } \mathbf{V}_o \text{ equals n.} \end{cases}$$

Example 6. The degree of the membership of the paper p_5 to the vague set $\mathbf{V}_{p_5}$ (see *Example 5*) is equal to 3/4.

It is clear that

Fact 10.
1. *Any vague set is a multiset in Blizard's sense* [1].
2. *Any vague set is a fuzzy set in Zadeh's sense* [28] *with μ as its membership function.*

5 Operations on Vague Sets

Let us denote by $\mathcal{V}$ the family of all vague sets determined by relations in the system $\Re(\Sigma)$. In the family $\mathcal{V}$ we can define a unary operation of the negation $\neg$ on vague sets, a union operation $\oplus$ and an intersection operation $\odot$ on any two vague sets.

Definition 13. Operations on vague sets.
Let $\mathbf{V_1} = \{R_i\}_{i \in I}$ and $\mathbf{V_2} = \{S_i\}_{i \in I}$ be vague sets determined by the sets $R \subseteq U$ and $S \subseteq U$, respectively. Then

(a) $\mathbf{V_1} \oplus \mathbf{V_2} = \{R_i\}_{i \in I} \oplus \{S_i\}_{i \in I} = \{R_i \cup S_i\}_{i \in I}$,
(b) $\mathbf{V_1} \odot \mathbf{V_2} = \{R_i\}_{i \in I} \odot \{S_i\}_{i \in I} = \{R_i \cap S_i\}_{i \in I}$,
(c) $\neg \mathbf{V_1} = \neg\{R_i\}_{i \in I} = \{U \setminus R_i\}_{i \in I}$.

The family $\mathbf{V_1} \oplus \mathbf{V_2}$ is called the *union* of the vague sets $\mathbf{V_1}$ and $\mathbf{V_2}$ determined by the relations R and S. The family $\mathbf{V_1} \odot \mathbf{V_2}$ is called the *intersection* of the vague sets $\mathbf{V_1}$ and $\mathbf{V_2}$ determined by the relations R and S. The family $\neg \mathbf{V_1}$ is called the *negation* of the vague set $\mathbf{V_1}$ determined by the relation R.

Theorem 1. *Let* $\mathbf{V_1} = \{R_i\}_{i\in I}$ *and* $\mathbf{V_2} = \{S_i\}_{i\in I}$ *be vague sets determined by the sets* R *and* S*, respectively. Then*

(a) $\underline{\mathbf{V_1} \oplus \mathbf{V_2}} = \underline{\mathbf{V_1}} \cup \underline{\mathbf{V_2}}$ *and* $\overline{\mathbf{V_1} \oplus \mathbf{V_2}} = \overline{\mathbf{V_1}} \cup \overline{\mathbf{V_2}}$*,*

(b) $\underline{\mathbf{V_1} \odot \mathbf{V_2}} = \underline{\mathbf{V_1}} \cap \underline{\mathbf{V_2}}$ *and* $\overline{\mathbf{V_1} \odot \mathbf{V_2}} = \overline{\mathbf{V_1}} \cap \overline{\mathbf{V_2}}$*,*

(c) $\underline{\neg\mathbf{V_1}} = U \setminus \overline{\mathbf{V_1}}$ *and* $\overline{\neg\mathbf{V_1}} = U \setminus \underline{\mathbf{V_1}}$*.*

Theorem 2. *The structure* $\mathfrak{B} = (\mathcal{V}, \oplus, \odot, \neg, \mathbf{0}, \mathbf{1})$ *is a Boolean algebra, where* $\mathbf{0} = \{\emptyset\}$ *and* $\mathbf{1} = \{U\}$*.*

We can easily observe that the above-defined operations on vague sets differ from Zadeh's operations on fuzzy sets, from standard operations in any field of sets and, in particular, from the operations on rough sets defined by Pomykała & Pomykała [23] and Bonikowski [2]. The family of all rough sets with operations defined in the latter two works is a Stone algebra.

6 On Logic of Vague Terms

How to solve the problem of logic of vague terms, logic of vague sentences (*vague logic*) based on the vague sets characterized in the previous sections? Answering this question requires a brief description of the problem of language representation of unit knowledge.

On the basis of our examples, let us consider two pieces of unit information about the paper p_5, with respect to the set R of all papers that have been referenced in at least 200 other papers:

first, exact unit knowledge

$$\overrightarrow{R_{ag}}(p_5) = \{p_2, p_3, p_5\}, \tag{ee}$$

next, vague unit knowledge:

$$\overrightarrow{R_{ag}}(p_5) = \mathit{VALUABLE}. \tag{e_1}$$

Let p_5 be the designator of the proper name a, R – the denotation (extension) of the name-predicate P ('*a paper that has been quoted in at least 200 other papers*'), and the vague name-predicate V ('*a paper which is valuable*') – a language representation of the vague quantity *VALUABLE*. Then a representation of the first equation (ee) is the logical atomic sentence

$$a \text{ is } P \tag{re}$$

and a representation of the second equation (e_1) is the vague sentence

$$a \text{ is } V. \tag{re_1}$$

In a similar way, we can represent, respectively, (ee) and (e_1) by means of a logical atomic sentence:

$$aP \text{ or } P(a), \tag{re'}$$

where P is the predicate (*'has been quoted in at least 200 other papers'*), and by means of a vague sentence

$$aV \text{ or } V(a), \tag{re'_1}$$

where V is the vague predicate (*'is valuable'*).

The sentence (re_1) (res. the sentence (re'_1)) is not a logical sentence, but it can be treated as a *sentential form*, which represents all logical sentences, in particular the sentence (re) (respectively sentence (re')) that arises by replacing the vague name-predicate (res. vague predicate) V by allowable sharp name-predicates (res. sharp predicates), whose denotations (extensions) constitute the vague set $\mathbf{V}_{p_5}$ being the denotation of V and, at the same time, the set of solutions to the equation (e_1) from the agent's point of view.

By analogy, we can consider every atomic vague sentence in the form $V(a)$, where a is an individual term and V – its vague predicate, as a *sentential form with V as a vague variable* running over all denotations of sharp predicates that can be substituted for V in order to get precise, true or false, logical sentences from the form $V(a)$. Then, the scope of the variable V is the vague set $\mathbf{V}_o$ determined by the designator o of the term a.

All the above remarks lead to a 'conservative', classical approach in searching for a logic of vague terms, or vague sentences, here referred to as *vague logic* (cf. Fine [9], Cresswell [7]). It is easy to see that all counterparts of laws of classical logic are laws of *vague logic* because, to name just one reason, vague sentences have an interpretation in Boolean algebra $\mathfrak{B}$ of vague sets (see Theorem 2).

We can distinguish two directions in seeking such a logic:

1a) all counterparts of tautologies of classical sentential calculus that are obtained by replacing sentence variables with atomic expressions of this logic (in the form $\mathcal{V}(x)$), representing vague atomic sentences (sentential functions in the form $V(a)$), are tautologies of *vague logic*,

1b) all counterparts of tautologies of classical predicate calculus that can be obtained by replacing predicate variables with vague predicate variables, representing vague predicates, are tautologies of *vague logic*;

2) *vague logic* should be a finite-valued logic, in which a value of any vague sentence $V(a)$ represented by its vague atomic expression (in the form $\mathcal{V}(x)$) is the multiplicity of membership of the designator o of a to the vague set $\mathbf{V}_o$ being the denotation of V, and the multiplicities of membership of the designators of the subjects of any composed vague sentence, represented by its composed vague formula, to the denotation (a vague set) corresponding to this sentence are functions of the multiplicities of membership of every designator of the subject of its atomic component to the denotation of its vague predicate.

It should be noticed that sentential connectives for *vague logic* should not satisfy *standard conditions* (see Malinowski [11]). For example, an alternative of two vague sentences $V(a)$ and $V(b)$ can be a 'true' vague sentence (sentential form) despite the fact that its arguments $V(a)$ and $V(b)$ are neither 'true' or 'false' sentential forms, i.e. in certain cases they represent true sentences, and in some other cases they represent false sentences. It is not contrary to the statement that all vague sentential forms which we obtain by a suitable substitution of sentential variables (resp. predicate variables) by vague sentences (resp. vague predicates) in laws of classical logic always represent true sentences. Thus they are laws of vague logic.

7 Final Remarks

1. The concept of vagueness was defined in the paper as an indefinite, vague quantity or property corresponding to the knowledge of an agent discovering a fragment of reality, and delivered in the form of the *equation of inexact knowledge of the agent*. A vague set was defined as a set (family) of all possible solutions (sets) of this equation and although our considerations were limited to the case of unary relations, they can easily be generalized to encompass any k-ary relations.
2. The idea of *vague sets* was derived from the idea of rough sets originating in the work of Zdzisław Pawlak, whose theory of rough sets takes a non-numerical, qualitative approach to the issue of vagueness, as opposed to the quantitative interpretation of vagueness provided by Lotfi Zadeh.
3. Vague sets, like rough sets, are based on the idea of a set approximation by two sets called the lower and the upper limits of this set. These two kinds of sets are families of sets approximated by suitable limits.
4. Pawlak's approach and the approach discussed in this paper both make a reference to the concept of a cognitive agent's knowledge about the objects of the reality being investigated (see Pawlak [20]). This knowledge is determined by the system of concepts that is determined by a system of their extensions (denotations). When the concept is vague, its denotation, in Pawlak's sense, is a rough set, while in the authors' sense – a vague set which, under some conditions, is a subset of the rough set.
5. In language representation, the *equation of inexact, vague knowledge of the agent* can be expressed by means of vague sentences containing a vague predicate. Its denotation (extension) is a family of all scopes of sharp predicates which, from the agent's viewpoint, can be substituted for the vague predicate. The denotation is, at the same time, the vague set of all solutions to the equation of the agent's vague knowledge.
6. Because vague sentences can be treated as sentential forms whose variables are vague predicates, all counterparts of tautologies of classical logic are laws of *vague logic* (logic of vague sentences).
7. *Vague logic* is based on classical logic but it is many-valued logic, because its sentential connectives are not extensional.

References

1. Blizard, W.D.: Multiset Theory. Notre Dame J. Formal Logic 30(1), 36–66 (1989)
2. Bonikowski, Z.: A Certain Conception of the Calculus of Rough Sets. Notre Dame J. Formal Logic 33, 412–421 (1992)
3. Bonikowski, Z.: Sets Approximated by Representations (in Polish, the doctoral dissertation prepared under the supervision of Prof. U.Wybraniec-Skardowska), Warszawa (1996)
4. Bonikowski, Z., Wybraniec-Skardowska, U.: Rough Sets and Vague Sets. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS, vol. 4585, pp. 122–132. Springer, Heidelberg (2007)

5. Bonissone, P., Tong, R.: Editorial: reasoning with uncertainty in expert systems. Int. J. Man–Machine Studies 22, 241–250 (1985)
6. Codd, E.F.: A Relational Model of Data for Large Shared Data Banks. Comm. ACM 13, 377–387 (1970)
7. Cresswell, M.J.: Logics and Languages. Methuen, London (1973)
8. Demri, S., Orłowska, E.: Incomplete Information: Structure, Inference, Complexity. Springer, Heidelberg (2002)
9. Fine, K.: Vagueness, Truth and Logic. Synthese 30, 265–300 (1975)
10. Iwiński, T.: Algebraic Approach to Rough Sets. Bull. Pol. Acad. Sci. Math. 35, 673–683 (1987)
11. Malinowski, G.: Many-Valued Logics. Oxford University Press, Oxford (1993)
12. Marcus, S.: A Typology of Imprecision. In: Brainstorming Workshop on Uncertainty in Membrane Computing Proceedings, Palma de Mallorca, pp. 169–191 (2004)
13. Marek, W., Pawlak, Z.: Rough Sets and Information Systems, ICS PAS Report 441 (1981)
14. Pagliani, P.: Rough Set Theory and Logic-Algebraic Structures. In: Orłowska, E. (ed.) Incomplete Information: Rough Set Analysis, pp. 109–190. Physica Verlag, Heidelberg (1998)
15. Parsons, S.: Current approaches to handling imperfect information in data and knowledge bases. IEEE Trans. Knowl. Data Eng. 8(3), 353–372 (1996)
16. Pawlak, Z.: Information Systems, ICS PAS Report 338 (1979)
17. Pawlak, Z.: Information Systems – Theoretical Foundations (in Polish). PWN – Polish Scientific Publishers, Warsaw (1981)
18. Pawlak, Z.: Information Systems – Theoretical Foundations. Information Systems 6, 205–218 (1981)
19. Pawlak, Z.: Rough Sets. Intern. J. Comp. Inform. Sci. 11, 341–356 (1982)
20. Pawlak, Z.: Rough Sets. Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht (1991)
21. Pawlak, Z.: Vagueness and uncertainty: A rough set perspective. Computat. Intelligence 11(2), 227–232 (1995)
22. Pawlak, Z.: Orthodox and Non-orthodox Sets - some Philosophical Remarks. Found. Comput. Decision Sci. 30(2), 133–140 (2005)
23. Pomykała, J., Pomykała, J.A.: The Stone Algebra of Rough Sets. Bull. Pol. Acad. Sci. Math. 36, 495–508 (1988)
24. Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press, Princeton (1976)
25. Skowron, A., Komorowski, J., Pawlak, Z., Polkowski, L.: Rough Sets Perspective on Data and Knowledge. In: Klösgen, W., Żytkow, J.M. (eds.) Handbook of Data Mining and Knowlewdge Discovery, pp. 134–149. Oxford University Press, Oxford (2002)
26. Słowiński, R., Stefanowski, J.: Rough-Set Reasoning about Uncertain Data. Fund. Inform. 23(2–3), 229–244 (1996)
27. Wybraniec-Skardowska, U.: Knowledge, Vagueness and Logic. Int. J. Appl. Math. Comput. Sci. 11, 719–737 (2001)
28. Zadeh, L.A.: Fuzzy Sets. Information and Control 8, 338–353 (1965)
29. Zadeh, L.A.: PRUF: A meaning representation language for natural languages. Int. J. Man–Machine Studies 10, 395–460 (1978)

Modified Indiscernibility Relation in the Theory of Rough Sets with Real-Valued Attributes: Application to Recognition of Fraunhofer Diffraction Patterns

Krzysztof A. Cyran

Institute of Informatics, Silesian University of Technology,
Akademicka 16, 44-100 Gliwice, Poland
krzysztof.cyran@polsl.pl

Abstract. The goal of the paper is to present the modification of classical indiscernibility relation, dedicated for rough set theory in a real-valued attributes space. Contrary to some other known generalizations, indiscernibility relation modified here, remains an equivalence relation and it is obtained by introducing a structure into collection of attributes. It defines real-valued subspaces, used in a multidimensional cluster analysis, partitioning the universe in a more natural way, as compared to one-dimensional discretization, iterated in classical model. Since the classical model is a special, extreme case of our modification, the modified version can be considered as more general. But more importantly, it allows for natural processing of real-valued attributes in a rough-set theory, broadening the scope of applications of classical, as well as variable precision rough set model, since the latter can utilize the proposed modification, equally well. In a case study, we show a real application of modified relation, a hybrid, opto-electronic recognizer of Fraunhofer diffraction patterns. Modified rough sets are used in an evolutionary optimization of the optical feature extractor implemented as a holographic ring-wedge detector. The classification is performed by a probabilistic neural network, whose error, assessed in an unbiased way is compared to earlier works.

Keywords: rough sets, indiscernibility relation, holographic ring-wedge detector, evolutionary optimization, probabilistic neural network, hybrid pattern recognition.

1 Introduction

In classical theory of rough sets, originated by Pawlak [32], the indiscernibility relation is generated by the information describing objects belonging to some finite set called universe. If this information is of discrete nature, than the classical form of this relation is natural and elegant notion. For many applications processing discrete attributes describing objects of the universe, such definition of indiscernibility relation is adequate, what implies that area of successful use of classical rough set methodology covers problems having natural discrete representation, consistent with granular nature of knowledge in this theory [32]. Such classical rough set model is particularly useful in automatic machine learning, knowledge acquisition and decision rules generation, applied to problems with discrete data not having enough size for application of statistical methods, demanding reliable estimation of distributions characterizing the underlying process [29,30].

If however, the problem is defined in a continuous domain, the classical indiscernibility relation almost surely builds one-element abstract classes, and therefore is not suitable for any generalization. To overcome this disadvantage, different approaches are proposed. The simplest is the discretization, but if this processes is iterated separately for single attributes, it induces artificial and highly nonlinear transformation of attribute space. Other approaches concentrate on generalization of the notion of indiscernibility relation into tolerance relation [25,36] or similarity relation [15,37,38]. The comparative study focused upon even more general approaches, assuming indiscernibility relation to be any binary reflexive relation, is given by Gomolinska [20]. Another interesting generalization of indiscernibility relation into characteristic relation, applicable for attributes with missing values (lost values or don't care conditions) is proposed by Grzymala-Busse [21,22].

In the paper we propose methodology, based on introduction of structure into a collection of conditional attributes, and treating certain groups defining this structure as multidimensional subspaces in a forthcoming cluster analysis. In this way we do not have to resign from equivalence relation, and at the same time, we obtain abstract classes uniting similar objects, belonging to the same clusters, in a continuous multidimensional space, as required by majority of classification problems.

Since the area of author's interests is focused in hybrid opto-electronic pattern recognition systems, the practical illustration of proposed modification concerns such system. However, with some exceptions, indicated at the end of section 2, the modification can find many more applications, especially, that it can be equally well adopted in a generalized variable precision rough set model, introduced by Ziarko [40], to meet requirements of analysis of huge data sets.

Automatic recognition of images constitutes an important area in the pattern recognition problems. Mait et al. [28], in a review article, state that "an examination of recent trends in imaging reveals a movement towards systems that balance processing between optics and electronics". Such systems are designed to perform heavy computations in optical mode, practically contributing no time delays, while post- processing is made in computers, often with the use of artificial intelligence (AI) methods. The foundations of one of such systems have been proposed by Casasent and Song [4], presenting the design of holographic ring wedge detectors (HRWD), and by George and Wang, who combined commercially available ring wedge-detector (RWD) and neural network (NN) in a one complete image recognition system [19]. Despite the completeness of the solution their system was of little practical importance, since commercially available

RWD was very expensive and moreover, could not be adapted to a particular problem. Casasent's HRWD, originally named by him as a computer generated hologram (CGH) had a lot of advantages over commercial RWD, most important being: much lower cost and adaptability. According to optical characteristics the HRWD belongs to a wider class of grating based diffractive optical variable devices (DOVDs) [11], which could be relatively easy obtained from computer generated masks, and are used for sampling the Fraunhofer diffraction pattern.

The pioneering works proposing the method of optimization of HRWD masks to a given application have been published by Cyran and Mrozek [10] and by Jaroszewicz et al. [23]. Mentioned method was successfully applied to a multi layer perceptron (MLP) based system, in a recognition of the type of subsurface stress in materials with embedded optical fiber [9,12,14]. Examples of application of the RWD-based feature extraction together with MLP-based classification module include systems designed by Podeszwa et al. [34] devoted for the monitoring of the engine condition, and by Jaroszewicz et al. [24] dedicated for airplane engines. Some other notable examples of applications of ring-wedge detectors and neural network systems, include works of Ganotra et al. [17] and Berfanger and George [3], concerning fingerprint recognition, face recognition [18], or image quality assessment [3]. The ring-wedge detector has been also used, as a light scatter detector, in a classification of airbone particles performed by Kaye et al. [26] and accurate characterization of particles or defects, present on or under the surface, useful in fabrication of integrated circuits, as presented by Nebeker and Hirleman [31]. The purely optical version of HRWD-MLP recognition system was considered by Cyran and Jaroszewicz [7], however, such system is limited by the development of optical neural networks. Simplified, to rings only, version of the device is reported by Fares et al. [16] to be applied in a rotation invariant recognition of letters. With all these applications, no wonder that Mait et al. [28] concluded: "few attempts have been made to design detectors with much consideration for the optics. A notable exception is ring-wedge detector designed for use in the Fourier plane of a coherent optical processor."

Obviously, MLP (or more generally any type of NN) is not the only classifier which could be applied for classification of patterns occurring in a feature space generated by HRWD. Moreover, the first version of optimization procedure favored the rough set based classifiers, due to identical (and therefore fully compatible) discrete nature of knowledge representation in the theory of rough sets applied both to HRWD optimization and to subsequent rough set based classification. The application of general ideas of obtaining such rough classifier was presented by Cyran and Jaroszewicz [8] and fast rough classifier implemented as PAL 26V12 element was considered and designed by Cyran [6]. Despite of inherent compatibility between optimization procedure and the classifier, the system remained sub optimal, because features extracted from HRWD generate continuous space, subject to unnatural discretization required by both: rough set based optimization and classifier.

Mentioned problems led to the idea, that in order to obtain the enhanced optimization method, the discretization required by classical indiscernibility relation in rough set theory, should be eliminated in such a way, which does not require the resignation from equivalence relation in a favor of some weaker form (like tolerance relation, for example). We achieved it by such modification of the indiscernibility relation which allows natural processing of real valued attributes. The paper presents this problem in the section 2. After focusing on indiscernibility relation related problems in section 2, section 3 starts with optical foundations of the recognition system considered, and it is followed by experimental results obtained from application of enhanced optimization methodology. The discussion and conclusions are included in section 4. Remarkably, the experimental application of modified indiscernibility relation into the system considered, improved the results of evolutionary optimization of holographic RWD and equivalently, enhanced the optimization of the HRWD generated feature space, dedicated for real-valued classifiers. It also gave theoretical basis for the latest design of two-way, neural network - rough set based classification system [5].

2 Modification of Indiscernibility Relation

Let us start with a brief analysis of the classical theory of rough sets, and the generalization of it, named the theory of rough sets with variable precision, in a context of data representation requirements. Next, the modification of indiscernibility relation is given. With modified indiscernibility relation, majority of notions defined in rough set theory (both in classical and generalized, variable precision form) can be naturally applied to the attributes having real-valued domain.

2.1 Analysis of Theory of Rough Sets with Discrete Attributes

The notion of a rough set has been defined for a representation, processing and understanding of imperfect knowledge. Such knowledge must be often sufficient in controlling, machine learning or pattern recognition. The rough approach is based on an assumption that each object is associated with some information, describing it, not necessarily, in an accurate and certain way. Objects described by the same information are not discernible. The indiscernibility relation, introduced here in an informal way, expresses the fact that theory of rough sets does not deal with individual objects, but with classes of objects which are indiscernible. Therefore the knowledge represented by classical rough sets is granular [32]. The simple consequence is that objects with natural real-valued representation, hardly match that scheme, and some preprocessing has to be performed, before such objects can be considered in a rough-set based frame. This preprocessing has the goal in making "indiscernible" objects which are close enough (but certainly discernible) in real-valued space. In majority of applications of rough set theory, this is obtained by subsequent discretization of all real-valued attributes. This, highly nonlinear process, is not natural and disadvantageous in many applications (such as an application presented in section 3). Before we present an alternative way of addressing the problem (in subsection 2.2), a formal definition of classical indiscernibility relation is given.

Let $S =< U, Q, v, f >$ be the information system composed of universe U, set of attributes Q, information function f, and a mapping v. This latter mapping associates each attribute $q \in Q$ with its domain V_q. The information function $f : U \times Q \rightarrow V$ is defined in such a way, that $f(x, q)$ reads as the value of attribute q for the element $x \in U$, and V denotes a domain of all attributes $q \in Q$ and is defined as a union of all domains of single attributes, i.e. $V = \bigcup_{q \in Q} V_q$. Then each nonempty set of attributes $C \subseteq Q$ defines the indiscernibility relation $I_0(C) \subseteq U \times U$ for $x, y \in U$ as

$$xI_0(C)y \Leftrightarrow \forall q \in C, f(x, q) = f(y, q). \tag{1}$$

Such definition, although theoretically applicable, both for discrete and continues domains V, is practically valuable only for discrete domains. For continuous domains such relation is too strong, because in practice all elements would have been discernible. Consequently, all abstract classes generated by I, would have been composed of exactly one element, what would have made the application of rough set theory notions possible, but senseless. The problem is that in the theory of rough sets, with each information system, we can associate some knowledge K_Q generated by the indiscernibility relation $I_0(Q)$; for continuous attributes the corresponding knowledge would have been too specific, to allow for any generalizations, required for classification of similar objects into common categories.

2.2 Indiscernibility Relation in Rough Sets with Real Valued Attributes

The consequence of the discussion ending the previous section is the need of discretization. If a problem is originally defined for real valued attributes, then before application of rough set theory, some clustering and discretization of continuous values of attributes should be performed. Let this process be denoted as a transformation described by a vector function $\mathbf{\Lambda} : \Re^{card(C)} \rightarrow \{1, 2, \ldots, \xi\}^{card(C)}$, where ξ is called the discretization factor. The discretization factor simply denotes the number of clusters covering the domain of each individual attribute $q \in C$. Theoretically, this factor could be different for different attributes, but without the loss of generality, we assume its constancy over the set of attributes. Then, the discretization of any individual attribute $q \in C$, can be denoted as a transformation defined by a scalar function $\mathit{\Lambda} : \Re \rightarrow \{1, 2, \ldots, \xi\}$. In this case, we obtain the classical form of indiscernibility relation, defined as

$$xI_0\left(\mathbf{\Lambda}[C]\right)y \Leftrightarrow \forall q \in C, f\left(x, \mathit{\Lambda}[q]\right) = f\left(y, \mathit{\Lambda}[q]\right). \tag{2}$$

Below, we will summarize, that majority (however, not all) of notions defined in a theory of rough sets *de facto* do not demand the strong version of indiscernibility relation I_0 defined by equation (1) (or by (2), if the discretization is required). From a formal point of view, what is really important, is the fact, that we assume the indiscernibility relation to be a relation of equivalence, i.e. it must be reflexive, symmetric and transitive. From practical point of view, objects indiscernible in a sense of rough set theory, should be such objects, which are close in a real-valued space. Any relation, having these properties, we denote by I, without any subscript, reserving subscripts for denoting particular forms of I. The exact form of I, defined as I_0 in (1) or (2), is not required, except for some notions, which we discuss later, for processing of the rough information. One can easily verify (by confrontation of the general form of indiscernibility relation I with presented below notions) that the following constructs form a logically consistent system, no matter what the specific form of the indiscernibility relation is. In particular it is true for such forms of this relation, which vary from classical form, both for discrete (1) and continuous (2) types of attributes, as presented below.

***C*-elementary sets.** Set Z is C-elementary, when all elements $x \in Z$ are C-indiscernible, i.e. they belong to the same class $[x]_{I(C)}$ of relation $I(C)$. If $C = Q$ then Z is elementary set in S. C-elementary set is therefore the atomic unit of knowledge about universe U with respect to C. Since C-elementary sets are defined by abstract classes of relation I, it follows that any equivalence relation can be used as I.

***C*-definable sets.** If a set X is a union of C-elementary sets then X is C-definable, i.e. it is definable with respect to knowledge K_C. A complement, a product, or an union of C-definable sets is also C-definable. Therefore the indiscernibility relation $I(C)$, by generating knowledge K_C , defines all what can be accurately expressed with the use of set of attributes C. Two information systems S and S' are equivalent if they have the same elementary sets. Then the knowledge K_Q is the same as knowledge $K_{Q'}$. Knowledge K_Q is more general than knowledge $K_{Q'}$ iff $I(Q') \subseteq I(Q)$, i.e. when each abstract class of the relation $I(Q')$ is included in some abstract class of $I(Q)$. C-definable sets, as unions of C-elementary sets are also defined for any equivalence relation I.

***C*-rough set *X*.** Any set being the union of C-elementary sets is a C-crisp set, any other collection of objects in universe U is called a C-rough set. A rough set contains a border, composed of elements such, that based on the knowledge generated by indiscernibility relation I, it is impossible to distinguish whether or not the element belongs to the set. Each rough set can be defined by two crisp sets, called lower and upper approximation of the rough set. Since C-crisp sets are unions of C-elementary sets, and C-rough set is defined by two C-crisp sets, therefore the notion of C-rough set is defined for any equivalence relation I, not necessarily for I_0.

***C*-lower approximation of rough set $X \subseteq U$.** The lower approximation of a rough set X is composed of those elements of universe, which belong *for sure* to X, based on indiscernibility relation I. Formally, C-lower approximation of a set $X \subseteq U$, denoted as $\underline{C}X$, is defined in the information system S, as $\underline{C}X = \{x \in U : [x]_{I(C)} \subseteq X\}$ and since it is a C-crisp set, it can be defined for arbitrary relation I.

***C*-upper approximation of rough set $X \subseteq U$.** The upper approximation of a rough set X is composed of those elements of universe, which *perhaps* belong to X, based on indiscernibility relation I. Formally, C-upper approximation of a set $X \subseteq U$, denoted as $\overline{C}X$ is defined in the information system S, as $\overline{C}X = \{x \in U : [x]_{I(C)} \cap X \neq \emptyset\}$ and since it is a C-crisp set, it can be defined for arbitrary relation I.

***C*-border of rough set *X*$\subseteq$*U*.** The border of a rough set is the difference between its upper and lower approximation. Formally, C-border of a set X, denoted as $Bn_C(X)$ is defined as $Bn_C(X) = \overline{C}X - \underline{C}X$, and as a difference of two C-crisp sets, its definition is based on arbitrary equivalence relation I.

Other notions, which are based on a notion of upper and/or lower approximation of a set $X \subseteq U$ with respect to a set of attributes C, include: C-positive region of the set $X \subseteq U$, C-negative region of the set $X \subseteq U$, sets roughly C-definable, sets internally C-undefinable, sets externally C-undefinable, sets totally C-undefinable, roughness of a set, C-accuracy of approximation of a set: $\alpha_C(X)$, C-quality of approximation of a set: $\gamma_C(X)$. An interesting comparison of this latter coefficient and Dempster-Shafer theory of evidence is given by Skowron and Grzymala-Busse [35].

Rough membership function of the element *x*:$\mu_{\mathbf{X}}^{\mathbf{C}}$*(x)*. The coefficient describing the level of uncertainty, whether the element $x \in U$ belongs to a set $X \subseteq U$ when indiscernible relation $I(C)$ generates the knowledge K_C in information system S, is a function denoted by $\mu_X^C(x)$ and defined as $\mu_X^C(x) = card\{X \cap [x]_{I(C)}\}/card\{[x]_{I(C)}\}$. This coefficient is also referred to as a rough membership function of an element x, due to similarities with membership function known from theory of fuzzy sets. This function gave base for the generalization of rough set theory called rough set model with variable precision [40]. This model assumes that lower and upper approximations are dependent on additional coefficient β, such that $0 \leq \beta \leq 0.5$, and are defined as $\underline{C}_\beta X = \{x \in U : \mu_X^C(x) \geq 1-\beta\}$ and $\overline{C}_\beta X = \{x \in U : \mu_X^C(x) > \beta\}$ respectively. The boundary in this model is defined as $Bn_C^\beta(X) = \{x \in U : \beta < \mu_X^C(x) < 1-\beta\}$. It is easy to observe that the classical rough set theory is the special case of variable precision model with $\beta = 0$. Since $\forall X \subseteq U$, $\underline{C}X \subseteq \underline{C}_\beta X \subseteq \overline{C}_\beta X \subseteq \overline{C}X$, variable precision model is a weaker form of theory as compared to classical model, and therefore it is often preferable in analysis of large information systems with some amount of contradicting data. The membership function of an element x can be also defined for a family of sets $\mathbf{X}$ as $\mu_{\mathbf{X}}^C(x) = card\{(\bigcup_{X_n \in \mathbf{X}} X_n) \cap [x]_{I(C)}\}/card\{[x]_{I(C)}\}$. If all subsets X_n of the family $\mathbf{X}$ are mutually disjoint, then $\forall x \in U$, $\mu_{\mathbf{X}}^C(x) = \Sigma_{X_n \in \mathbf{X}}\ \mu_{X_n}^C(x)$. Since the definition of the rough membership function of the element $\mu_X^C(x)$ assumes only the existence of classes of equivalence of the relation I, and the variable precision model formally differs from classical model only in the definition of lower and upper approximation with the use of this coefficient, therefore all presented above notions are defined for arbitrary I also in this generalized model.

Notions of a rough set theory, applicable for a separate set X, are generally applicable also for families of sets $\mathbf{X} = \{X_1, X_2, \ldots, X_N\}$, where $X_n \subseteq U$, and $n = 1, \ldots, N$. The lower approximation of a family of sets is a family of lower approximations of sets belonging to family considered. Formally, $\underline{C}\mathbf{X} = \{\underline{C}X_1, \underline{C}X_2, \ldots, \underline{C}X_N\}$. As a family of C-crisp sets, the definition of C-lower approximation of family of sets is based on arbitrary relation of equivalence I. Similarly, C-upper approximation of family of sets is a family of upper approximations of sets belonging to family considered. Formally, $\overline{C}\mathbf{X} = \{\overline{C}X_1, \overline{C}X_2, \ldots, \overline{C}X_N\}$. This notion is valid for any relation of equivalence I, for reasons identical to those, presented for C-lower approximation of family of sets.

Other notions, which are based on a notion of upper and/or lower approximation of a family of sets X, with respect to a set of attributes C, include: C-border of family of sets, C-negative region of the family of sets, C-negative region of the family of sets, C-accuracy of approximation of a family of sets, C-quality of approximation of a family of sets. This latter coefficient is especially interesting for the application presented in the subsequent section, since it is used as an objective function in a procedure of optimization of the feature extractor. For this purpose, the considered family of sets is a family of abstract classes generated by the decision attribute d being the class of the image to be recognized (see section 3). Here, we define this coefficient for any family of sets $\mathbf{X}$, as $\gamma_C(\mathbf{X}) = card[Pos_C(\mathbf{X})]/card(U)$.

Conclusion. The analysis of above notions indicates, that they do not require any particular form of the indiscernibility relation (like for example the classical form referred to as I_0). They are defined for any form of the indiscernibility relation (satisfying reflexity, symmetry and transitiveness), denoted by I and are strict analogs of classical notions defined with the assumption of original form of indiscernibility relation I_0 defined in (1) and (2). Therefore, the exact form of the indiscernibility relation, as proposed by classical theory of rough sets, as well as by its generalization named variable precision model, is not actually required for presented notions to create a coherent logical system. Some papers, referred in introduction, go further in this generalizing tendency, resigning from the requirement of equivalence relation; working with such generalizations, however, is often not natural in problems, such as classification, when notion of abstract classes, inherently involved in equivalence relation, is of great importance. Therefore, we propose such modification of indiscernibility relation, which is particularly useful in pattern recognition problems, dealing with a space of continuous attributes and defined in terms of equivalence relation.

To introduce formally the modification, let us change the notation of indiscernibility relation as being now dependent on a family of sets of attributes, instead of being dependent simply on a set of attributes. By the family of sets of attributes, we understand a subset of a power set, based on the set of attributes, such, that all elements of this subset (these elements are subsets of the set of attributes) are mutually disjoint, and their union is equal to the considered set of attributes. This allows us to introduce some structure into, originally unstructured, set of attributes, which the relation depends on [13]. Let $\mathbf{C} = \{C_1, C_2, \ldots, C_N\}$ be introduced above family of disjoint sets of attributes $C_n \subseteq Q$ such that unstructured set of attributes $C \subseteq Q$ is equal to the union of members of the family $\mathbf{C}$, i.e. $\mathbf{C} = \bigcup_{C_n \in \mathbf{C}} C_n$. Then, let the indiscernibility relation be dependent on $\mathbf{C}$ instead of being dependent on C. Observe that both $\mathbf{C}$ and C contain the same collection of single attributes, however $\mathbf{C}$ includes additional structure as compared to C. If this structure is irrelevant for the problem considered, it can be simply ignored and we can obtain, as a special case, the classical version of indiscernibility relation I_0. However we can also obtain other versions of this modified relation for which the introduced structure is meaningful. Let relation $I_1(C) \subseteq U \times U$ be such form of a relation I which is different from I_0

$$xI_1(\mathbf{C})y \Leftrightarrow \forall C_n \in \mathbf{C}, Clus(x, C_n) = Clus(y, C_n). \tag{3}$$

where $x, y \in U$, and $Clus(x, C_n)$ denotes the number of the cluster, that the element x belongs to. The cluster analysis is therefore required to be performed in a continuous vector spaces defined by sets of real valued conditional attributes $C_n \in \mathbf{C}$. There are two extreme cases of this relation, obtained when family $\mathbf{C}$ is composed of exactly one set of conditional attributes C, and when family $\mathbf{C}$ is composed of $card(C)$ sets, each containing exactly one conditional attribute $q \in C$.

The classical form I_0 of the indiscernibility relation is obtained as the latter extreme special case of modified version I_1, because then clustering and discretization is performed separately for each continuous attribute. Formally, it can be written as

$$I_0\left(\mathbf{\Lambda}[C]\right) \equiv I_1(\mathbf{C}) \Leftrightarrow \mathbf{C} = \left\{ \{q_n\} : C = \bigcup_{q_n \in C} \{q_n\} \right\} \wedge Clus\left(x, \{q_n\}\right) = f\left(x, \Lambda[q_n]\right). \tag{4}$$

In other words, the classical form I_0 of the indiscernibility relation can be obtained as a special case of modified version I_1 if we assume that family $\mathbf{C}$ is composed of such subsets C_n, that each contains just one attribute, and the discretization of each continuous attribute is based on separate cluster analysis as required by a scalar function Λ applied to each of attributes q_n.

Here we discuss some of the notions of rough set theory that cannot be used in a common sense with the modified indiscernibility relation. We start with so called basic sets which are abstract classes of relation $I(\{q\})$ defined for singe attribute q. These are simply sets composed of elements indiscernible with respect to single attribute q. Obviously, this notion loses its meaning when I_1 is used instead of I_0, because abstract classes generated by $I_0(\{q\})$ are always unions of some abstract classes generated by $I_0(C)$, however abstract classes generated by $I_1(\{q\})$ not necessarily are unions of abstract classes generated by $I_1(C)$. Therefore the conclusion that knowledge $K_{\{q\}}$ generated by $I_0(\{q\})$ is always more general than knowledge K_C generated by $I_0(C)$, no longer holds when I_1 is used instead of I_0. Similarly, notions of reducts, relative reducts, cores and relative cores no longer are applicable in their classical sense, since their definitions are strongly associated with single attributes. Joining these attributes into members of family $\mathbf{C}$, destroys individual treatment of attributes, required for these notions to have their well known meaning. However, as long as rough set theory usage in continuous attribute space, does not exceed the collection of notions described ahead of the definition (3), the modified I_1 version should be considered more advantageous, as compared to the classical form I_0. In particular, this is true in processing of knowledge obtained from the holographic ring wedge detector, when the quality of approximation of family of sets plays the major role. We present this application as an illustrative example.

3 Application to Fraunhofer Pattern Recognizer

Presented below system belongs to a class of fast hybrid opto-electronic pattern recognizers. The feature extraction subsystem is processing the information optically. Let us start a description of such feature extractor by giving a physical basis, required to understand the properties of feature vectors generated by this subsystem, followed by the description of enhanced method of HRWD optimization and experimental results of the usage of this method. This illustrative section is completed with the description of probabilistic neural network (PNN) based classifier and experimental results of the application of it into Fraunhofer pattern recognition.

3.1 Optical Foundations

In homogeneous and isotropic medium which is free of charge ($\rho = 0$) and currents ($j = 0$) Maxwell equations result in a wave equation

$$\Delta^2 \mathbf{G} - \epsilon' \mu' \frac{\partial^2 \mathbf{G}}{\partial t^2} = 0. \tag{5}$$

where $\mathbf{G}$ denotes electric ($\mathbf{E}$) or magnetic ($\mathbf{H}$) field, and a product $\epsilon' \mu'$ is the reciprocal of squared velocity of a wave in a medium. Application of this equation to a space with obstacles like apertures or diaphragms should result in equations describing the diffraction of the light at these obstacles. However the solution is very complicated for special cases and impossible for the general case. Therefore the simplification should be used which assumes a scalar field u instead of vector field $\mathbf{G}$. In such a case the information about the light polarization is lost and it holds that

$$\nabla^2 u - \frac{1}{\nu^2} \frac{\partial^2 u}{\partial t^2} = 0. \tag{6}$$

Simplified in this way theory, called the scalar Kirchhoff's theory, describes the diffraction of the light at various obstacles. According to this theory, scalar complex amplitude $u_0(P)$ of a light oscillation, caused by the diffraction, is given in a point of observation P by the Kirchhoff's integral [33]

$$u_0(P) = \frac{1}{4\pi} \int\limits_{\Sigma} \left[\frac{e^{ikr}}{r} \frac{du_0}{dn} - u_0 \frac{d}{dn} \left(\frac{e^{ikr}}{r} \right) \right] d\Sigma. \tag{7}$$

where Σ denotes closed surface with point P and without the light source, n is an external normal to the surface Σ, $k = 2\pi/\lambda$ is a propagation constant,

u_0 denotes scalar amplitude on a surface σ, and r is the distance between any point covered inside surface Σ to the observation point P. Formula (7) states that amplitude u_0 in point P does not depend on the state of oscillations in the whole area surrounding this point (what would result from Huygens theory) but, depends only on state of oscillations on a surface Σ. All other oscillations inside this surface are canceling each other. Application of Kirchhoff's theorem to a diffraction on a flat diaphragm with aperture of any shape and size gives the integral stretched only on a surface Σ_A covering the aperture. Such integral can be transformed to [33]

$$u_0(P) = -\frac{ik}{4\pi}\int\limits_{\Sigma_A} u_0(1+\cos\theta)\frac{e^{ikr}}{r}d\Sigma_A. \tag{8}$$

where θ denotes an angle between radius r from any point of aperture to point of observation, and the internal normal of the aperture.

Since any transparent image is, in fact, a collection of diaphragms and apertures of various shapes and sizes, therefore such image, when illuminated by coherent light, generates the diffraction pattern, described in scalar approximation by the Kirchhoff's integral (7). Let coordinates of any point A, in an image plane, are denoted by (x, y), and let an amplitude of light oscillation in this point, be $\mu(x, y)$. Furthermore, let coordinates (ξ, η) of an observation point P be chosen as

$$\xi = \frac{2\pi}{\lambda}\sin\theta, \eta = \frac{2\pi}{\lambda}\sin\varphi. \tag{9}$$

where: λ denotes the length of the light wave, whereas θ and φ are angles between the radius from the point of observation P to point A, and planes (x, z) and (y, z), respectively. These planes are two planes of such coordinate system (x, y, z), whose axes x and y are in the image plane, and axis z is perpendicular to the image plane (it is called optical axis). Let coordinate system (x', y') be the system with the beginning at point P and such that its plane (x', y') is parallel to the plane of the coordinate system (x, y). It is worth to notice, that coordinates of one particular point in the observation system (ξ, η) correspond to coordinates of all points P of the system (x', y'), such that the angles between axis z and a line connecting these points with some points A of the plane (x, y), are θ and φ, respectively. In other words, all radii AP, connecting points A of the plane (x, y) and points P of the plane (x', y'), which are parallel to each other, are represented in a system (ξ, η) by one point. Such transformation of the coordinate systems is physically obtained in the back focal plane of the lens, placed perpendicularly to the optical axis z. In this case, all parallel radii represent parallel light beams, diffracted on the image (see Fig. 1) and focused in the same point in a focal plane. Moreover, the integral (7), when expressed in a coordinate system (ξ, η), can be transformed to [33]

$$u_0(\xi, \eta) = \frac{1}{2\pi}\int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty} \nu(x, y)e^{-i(\xi x+\eta y)}dxdy. \tag{10}$$

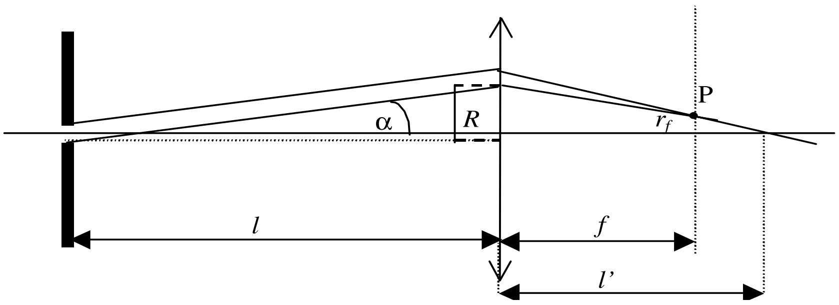


Fig. 1. The operation of the spherical lens

Geometrical relationships (Fig. 1) reveal that

$$r_f = R\frac{l' - f}{l'}. \tag{11}$$

On the other hand the operation of the lens is given by

$$\frac{1}{f} = \frac{1}{l} + \frac{1}{l'}. \tag{12}$$

Putting this equation to (11), after elementary algebra, we obtain

$$\frac{R}{l} = \frac{r_f}{f}. \tag{13}$$

Since angles θ and φ (corresponding to angle α in Fig. 1, in a plane (x, z) and (y, z), respectively) are small, therefore equations (9), having in mind (13), can be rewritten as

$$\xi = \frac{2\pi}{\lambda}\frac{x_f}{f}, \eta = \frac{2\pi}{\lambda}\frac{y_f}{f} \tag{14}$$

where x_f and y_f denote Cartesian coordinates in a focal plane of the lens. Equation (10) expressed in these coordinates can be written as

$$u_0(x_f, y_f) = \frac{1}{2\pi}\int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty} \nu(x, y)e^{-i2\pi\left(\frac{x_f}{\lambda f}x+\frac{y_f}{\lambda f}y\right)}dxdy. \tag{15}$$

Setting new coordinates (u, v) as

$$u = \frac{x_f}{\lambda f}, v = \frac{y_f}{\lambda f} \tag{16}$$

we have finally the equation

$$u_0(u, v) = \frac{1}{2\pi}\int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty} \nu(x, y)e^{-i2\pi(ux+vy)}dxdy. \tag{17}$$

which is (up to the constant factor k) a Fourier integral. This is essentially the Fraunhofer approximation of Kirchhoff's integral, and is also referred to as a Fraunhofer diffraction pattern [27]. The complex amplitude of the Fraunhofer diffraction pattern obtained in a back focal plane of the lens is therefore a Fourier transform of the complex amplitude from the image plane

$$u_0(u,v) = k\mathcal{F}\{\nu(x,y)\}. \tag{18}$$

This fact is very often used in a design of hybrid systems for recognition of images in a spatial frequency domain. One prominent example is the system with a feature extractor built as a HRWD placed in a back focal plane of the lens. The HRWD itself consists of two parts: a part composed of rings R_i and a part containing wedges W_j. Each of elements R_i or W_j is covered with a grating of particular spatial frequency and orientation, so that the light, passing through the given region, is diffracted and focused by some other lens, at certain cell of array of photodetectors. The photodetector, in turn, integrates the intensity of the light and generates one feature used in classification.

3.2 Enhanced Optimization Method

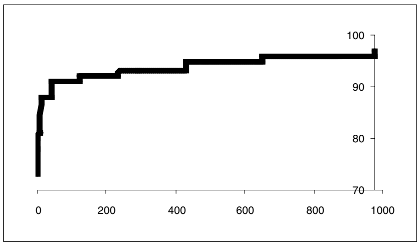
The system considered above can be used for the recognition of images invariant with respect to translation, rotation and size, based on the properties of Fourier transform and the way of sampling the Fraunhofer diffraction pattern by the HRWD. Standard HRWD based feature extractor can be optimized to obtain even better recognition properties of the system. To perform any optimization one needs the objective function and the method of search in a space of solutions. These two problems are discussed wider below.

Let ordered 5-tuple $T =< U, C, \{d\}, v, f >$ be the decision table obtained from the information system $S =< U, Q, v, f >$ by a decomposition of the set of attributes Q into two mutually disjoint sets: the set of conditional attributes C and the set $\{d\}$ composed of one decision attribute d. Let each conditional attribute $c \in C$ be one feature obtained from HRWD, and let decision attribute d be the number of the class to be recognized. Obviously the domain of any of such conditional attributes is $\Re$ and the domain of decision attribute d is a subset of first natural numbers, with cardinality equal to the number of recognized classes. Furthermore, let $\mathbf{D} = \{[x_n]_{I_0(\{d\})} : x_n \in U\}$ be the family of such sets of images where each set contains all images belonging to the same class. Observe that the classical form of the indiscernibility relation I_0 is used in this definition, due to discrete nature of the domain of decision attribute d.

Based on the results of discussion given by Cyran and Mrozek [10], we argue that the rough set based coefficient, called quality of approximation of family $\mathbf{D}$ by conditional attributes belonging to C, and denoted by $\gamma_C(\mathbf{D})$, is a good objective function in the optimization of feature extractor in problems with multimodal distribution of classes in a feature space. This is so, because this coefficient indicates the level of determinism of the decision table, what in turn, is relevant for the classification. On the other hand, based on the conclusion given in 2.2, in the case of real valued attributes C, the preferred form of indiscernibility relation, being so crucial for rough set theory in general (and therefore for the computation of $\gamma_C(\mathbf{D})$ objective in particular), is the form defined by (3). Therefore the optimization with the objective function $\gamma_C(\mathbf{D})$ computed with respect to classical form of indiscernibility relation for real valued attributes C given in (2) produces sub-optimal solutions. This drawback can be eliminated if modified version proposed in (3) is used instead of classical form defined in (2). However the generalized form (3) requires the definition of some structure in a set of conditional attributes. This is task dependent, and in our case the architecture of the feature extractor having different properties of wedges and rings, defines natural structure, as a family $\mathbf{C} = \{C_R, C_W\}$, composed of two sets: a set of attributes corresponding to rings C_R, and a set of attributes corresponding to wedges C_W. With this structure introduced into set of conditional attributes, the coefficient $\gamma_C(\mathbf{D})$ computed with respect to modified indiscernibility relation (3), is en enhanced objective function for optimization of the HRWD.

Since defined above enhanced objective function is not differentiable, gradient-based search method should be excluded. However the HRWD can be optimized in a framework of evolutionary algorithm. The maximum value of fitness 97%, having the meaning of $\gamma_C(D*) = 0.97$, was obtained in 976 generation for population composed of 50 individuals (Fig. 2). The computer generated mask of optimal HRWD, named $\mathbf{x_{opt}}$ is designed for a system with a coherent light wave length λ=635nm, emitted by laser diode and for a lens L with a focal length f_L=1m. In order to keep the resolution capability of the system, the diameter of the HRWD in a Fourier plane should be equal to the diameter of the Airy disc given by: $s_{HRWD} = 4 \times 1.22 \times \lambda \times f_{L1}/s_{min} = 2.07$mm, if the assumed minimum size of recognizable objects is given by $s_{min} = 1.5$mm. Assuming also the rectangular array of photodetectors of the size s=5mm, forming four rows (i=1,...,4) and four columns ($j = 1, \dots, 4$), and setting the distance in vertical direction from the optical axis to the upper edge of the array $H = 50$mm we obtain values of angles θ_{ij} presented in a Table 1. Similar results for the distances d_{ij} are in Table 2.

Since the software for generating HRWD masks has been designed in a such way, that distances d_{ij} are given in units equal to a one-tenth of a percent of the

a) 

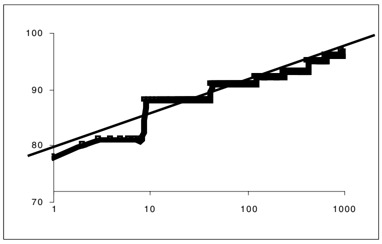
b) 

Fig. 2. Process of evolutionary optimization of HRWD. The courses present the fitness of $\mathbf{x_{opt}}$ expressed in percents: a) linear scale, b) logarithmic horizontal scale.

Table 1. The values of angles θ_{ij} (expressed in degrees) defining the HRWD gratings

4	3	2	1	$\leftarrow j,\ i \downarrow$
20.22	14.74	8.97	3.01	1
22.38	16.39	10.01	3.37	2
25.02	18.43	11.31	3.81	3
28.30	21.04	12.99	4.40	4

Table 2. Distances d_{ij} between striae [μm]

4	3	2	1	$\leftarrow j,\ i \downarrow$
12.54	12.93	13.20	13.35	1
13.82	14.33	14.71	14.92	2
15.34	16.06	16.60	16.90	3
17.20	18.24	19.04	19.48	4

Table 3. Distances d_{ij} between striae, in units used by software generating HRWD masks

4	3	2	1	$\leftarrow j,\ i \downarrow$
12.14	12.52	12.78	12.92	1
13.38	13.88	14.24	14.44	2
14.86	15.55	16.08	16.36	3
16.65	17.65	18.43	18.86	4

radius of HRWD, therefore for $R_{HRWD} = s_{HRWD}/2 = 1.035$mm, we give in a Table 3 the proper values, expressed in these units.

3.3 PNN Based Classification

In our design the input layer of the probabilistic neural network (PNN) used as a classifier is composed of N elements to process N-dimensional feature vectors generated by HRWD ($N = N_R + N_W$). The pattern layer consists of M pools of pattern neurons, associated with M classes of intermodal interference to be recognized. We used in that layer RBF neurons with Gaussian transfer function, being the kernel function. Then, the width of the kernel function is simply a standard deviation σ of the Gaussian bell. Each neuron of pattern layer is connected with every neuron of input layer and the weight vectors of pattern layer are equal to feature vectors present in a training set. Contrary to the pattern layer, the summation layer consisting of M neurons, is organized in a such way, that only one output neuron is connected with neurons from any summation layer pool. When using such networks as classifiers, formally, there is a need to multiply the output values by prior probabilities P_j. However in our cases, all priors are equal and therefore, results can be obtained directly on outputs of the network.

We verified the recognition abilities by a classification of speckle structure images, obtained from the output of the optical fiber. The experiments were conducted for a set of 128 images of speckle patterns generated by intermodal interference occurring in optical fiber and belonging to eight classes taken in 16 sessions S_l ($l = 1, \ldots, 16$). The Fraunhofer diffraction patterns of input images were obtained by calculating the intensity patterns from discrete Fourier transform equivalent to (17). The training set consisted of 120 images, taken out in 15 sessions, and the testing set contained 8 images, belonging to different classes, representing one session S_l. The process of training and testing was performed 16 times, according to *delete*-8 jackknife method, i.e., for each iteration, another session composed of 8 images was used for the testing set, and all but one sessions were used for the training set. That gave the basis for reliable cross-validation with still reasonable number of images used for training, and the reasonable computational time. This time was eight times shorter, as compared to classical leave-one-out method, which, for all discussions in a paper is equivalent to *delete*1 jackknife method, since the only difference, the resubstitution error of a prediction model, is not addressed in a paper. Jackknife method was used for cross validation of PNN results, because of unbiased estimation of true error in probabilistic classification (contrary to underestimated error - however having smaller variance obtained by Bootstrap method) [1,39]. Therefore, choice of *delete*-8 jackknife method, was a sort of tradeoff between accuracy (standard deviation of estimated normalized decision error was 0.012), unbiased estimate of the error, and computational effort. The results of such testing of the PNN applied to classification of images in a feature space obtained from a standard, optimized, and optimized with modified indiscernibility relation HRWDs, are presented in Table 4. More detailed results of all jackknife tests are presented in Table 5, Fig. 3 and Fig. 4.

The normalized decision errors, ranging from 1.5 to 2 percent, indicate good overall recognition abilities of the system. The 20% reduction of this error is obtained by optimization of HRWD with classical indiscernibility relation. Further 6% error reduction, is caused solely by a modification of indiscernibility relation, according to (3).

Table 4. Results of testing the classification abilities of the system. The classifier is a PNN having Gaussian radial function with standard deviation $\sigma = 0.125$. In the last column the improvement is computed with respect to Standard HRWD (first value) and with respect to HRWD optimized with standard indiscernibility relation (value in a parentheses).

	Correct decisions [%]	Normalized decision error [%]	Improvement [%]
Standard HRWD	84.4	1.95	0.0 (-25.0)
HRWD optimized with standard indiscernibility relation	87.5	1.56	20.0 (0.0)
HRWD optimized with modified indiscernibility relation	88.3	1.46	25.1 (6.4)

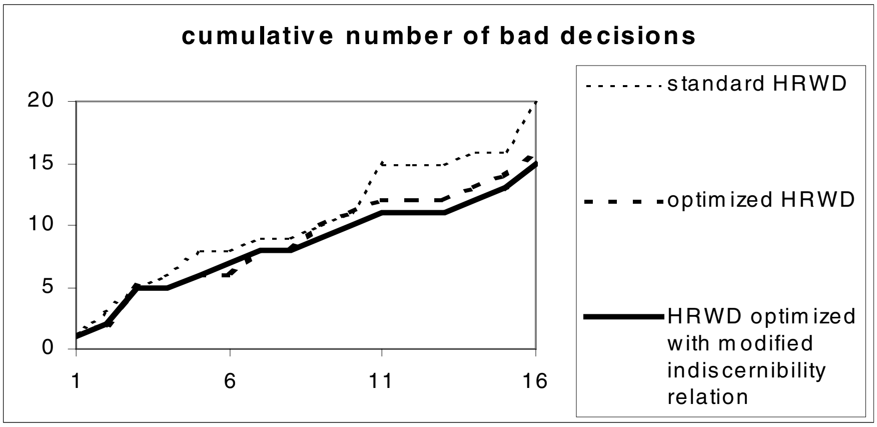


Fig. 3. Results of testing the HRWD-PNN system. The horizontal axis represents the number of the test, the vertical axis is a cumulative number of bad decisions. Starting from test 9 to the end, the cumulative number of bad decisions is better for optimization of HRWD with modified indiscernibility relation, as compared to optimization with classical version of this relation.

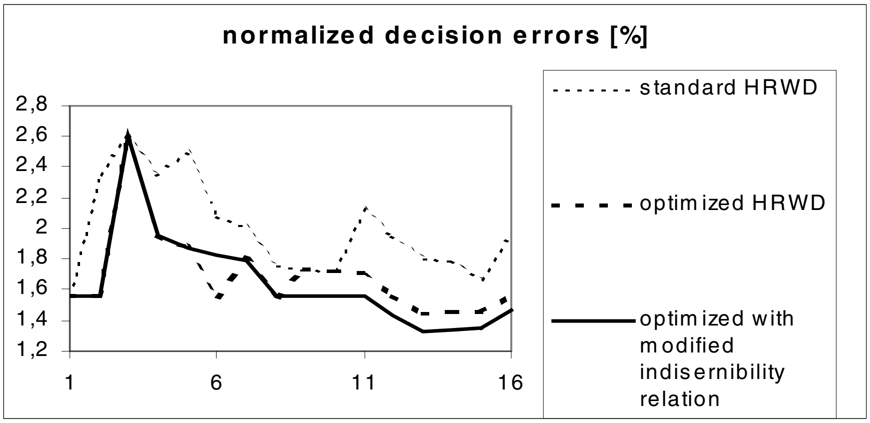


Fig. 4. Results of testing the HRWD-PNN system. The horizontal axis represents the number of the test, while the vertical axis is a normalized decision error averaged over tests from the first to given, represented by the value of horizontal axis. Observe, that for averaging over more than 8 tests, the results for recognition with HRWD optimized with modified indiscernibility relation are outperforming both: results for HRWD optimized with classical version of indiscernibility relation and results for standard HRWD.

In order to understand the scale of this improvement, not looking too impressive at first glance, one should refer to a Fig. 2 and take into consideration, that this additional 6% error reduction is obtained over an already optimized

Table 5. Results of PNN testing for tests number 1 to 16. Bold font is used for results differing between optimization with standard and modified version of indiscernibility relation. Bold underlined results indicate improvement when modified relation is used instead of classical. Bold results without underlining indicate the opposite.

NUMBER OF TEST SESSION:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	NUMBER OF BAD DECISIONS															
Standard HRWD	1	2	2	1	2	0	1	0	1	1	4	0	0	1	0	4
optimized with standard indiscernibility relation	1	1	3	0	1	0	2	0	2	1	1	0	0	1	1	2
optimized with modified indiscernibility relation	1	1	3	0	1	1	1	0	1	1	1	0	0	1	1	2

solution. The level of difficulty can be grasped observing that, on average, the increase of the objective function is well mimicked by a straight line, if a generation number axis is drawn in a log scale. This means, that the growth of objective is, on average, well approximated by a logarithmic function of the generation number. It experimentally reflects a well known fact, stating that, the better current solution is, the harder is to optimize it further (harder, means: it requires more generations in evolutionary process).

4 Discussion and Conclusions

The paper presents a modification of the indiscernibility relation, used in the theory of rough sets. This theory has been successfully applied to many machine learning and artificial intelligence oriented problems. However, it is well known limitation of this theory, that it processes continuous attributes in an unnatural way. To support more natural processing, the modification of indiscernibility relation has been proposed (3), such that the indiscernibility relation remains the equivalence relation, but the processing of continuous attributes becomes more natural. This modification introduces the information about structure into unstructured in classical version collection of attributes that the relation is dependent on. It has been shown that the classical relation is the special case of the modified version, therefore proposed modification can be recognized as being more general (yet, not as general, as indiscernibility relations, which are no longer equivalence relations). Remarkably, proposed generalization is equivalently valid for classical theory of rough sets, as well as for the variable precision model, predominantly used in machine learning applied to huge data sets.

Proposed in a paper modification of indiscernibility relation, introduces the flexibility in definition of particular special case, which is most natural to given application. In the case of real-valued attributes, our modification allows for performing multidimensional cluster analysis, contrary to multiple one-dimensional analyses, required by the classical form. In majority of cases, the cluster analysis should be performed in a space, generated by all attributes. This corresponds to a family $\mathbf{C}$ composed of one set ($card(\mathbf{C}) = 1$), containing all conditional attributes, and it is the opposite case, as compared to the classical relation, assuming that family $\mathbf{C}$ is composed of one-element disjoint sets, and therefore, satisfying equation $card(\mathbf{C}) = card(C)$. However, other less extreme cases are allowed as well and, in an experimental study, we use a family $\mathbf{C} = \{C_R, C_W\}$, composed of two sets containing 8 elements, each. Such structure seems to be natural for application having two-way architecture, like HRWD based feature extractor.

Presented modification has been applied in optimization procedure of the hybrid opto-electronic pattern recognition system composed of HRWD and PNN. It allowed to improve the recognition abilities by reducing the normalized decision error by 6.5%, if a system, optimized with classical indiscernibility relation, is treated as the reference. One should notice, that this improvement is achieved with respect to a reference, being already optimized solution, which makes any further improvement difficult. Obtained results experimentally confirm our claims concerning sub optimality of earlier solutions. Presented experiment is an illustration of application of proposed methodology into hybrid pattern recognizer. However, we think, that presented modification of indiscernibility relation will find many more applications in rough set based machine learning, since it gives natural way of processing real valued attributes, within a rough set based formalism.

Certainly there are also limitations. Because some known in rough set theory notions loose their meaning, when modified relation is to be applied, therefore, if for any reason, they are supposed to play relevant role in a problem, the proposed modification can be hardly applied in any other than classical special case form. One prominent example concerns so called basic sets in a universe U, defined by the indiscernibility relation, computed with respect to single attributes, as opposed to modified relation predominantly designed to deal with sets of attributes defining a vector space, used for common cluster analysis. This modification is especially useful in the case of information systems with real valued conditional attributes representing vector space $\Re^N$, such as systems of non syntactic pattern recognition. The experimental example belongs to this class of problems and illustrates the potential of modified indiscernibility relation for processing real-valued data in a rough set based theory.

References

1. Azuaje, F.: Genomic data sampling and its effect on classification performance assessment. BMC Bioinformatics 4(1), 5–16 (2003)
2. Berfanger, D.M., George, N.: All-digital ring-wedge detector applied to fingerprint recognition. App. Opt. 38(2), 357–369 (1999)
3. Berfanger, D.M., George, N.: All-digital ring wedge detector applied to image quality assessment. App. Opt. 39(23), 4080–4097 (2000)
4. Casasent, D., Song, J.: A computer generated hologram for diffraction-pattern sampling. Proc. SPIE 523, 227–236 (1985)

5. Cyran, K.A.: Integration of classifiers working in discrete and real valued feature space applied in two-way opto-electronic image recognition system. In: Proc. of IASTED Conference: Visualization, Imaging & Image Processing, Benidorn, Spain (accepted, 2005)
6. Cyran, K.A.: PLD-based rough classifier of Fraunhofer diffraction pattern. In: Proc. Int. Conf. Comp. Comm. Contr. Tech., Orlando, pp. 163–168 (2003)
7. Cyran, K.A., Jaroszewicz, L.R.: Concurrent signal processing in optimized hybrid CGH-ANN system. Opt. Appl. 31, 681–689 (2001)
8. Cyran, K.A., Jaroszewicz, L.R.: Rough set based classifiction of interferometric images. In: Jacquot, P., Fournier, J.M. (eds.) Interferometry in Speckle Light. Theory and Applictions, pp. 413–420. Springer, Heidelberg (2000)
9. Cyran, K.A., Jaroszewicz, L.R., Niedziela, T.: Neural network based automatic diffraction pattern recognition. Opto-elect. Rev. 9, 301–307 (2001)
10. Cyran, K.A., Mrozek, A.: Rough sets in hybrid methods for pattern recognition. Int. J. Intell. Sys. 16, 149–168 (2001)
11. Cyran, K.A., Niedziela, T., Jaroszewicz, L.R.: Grating-based DOVDs in high-speed semantic pattern recognition. Holography 12(2), 10–12 (2001)
12. Cyran, K.A., Niedziela, T., Jaroszewicz, J.R., Podeszwa, T.: Neural classifiers in diffraction image processing. In: Proc. Int. Conf. Comp. Vision Graph., Zakopane, Poland, pp. 223–228 (2002)
13. Cyran, K.A., Stanczyk, U.: Indiscernibility relation for continuous attributes: Application in image recognition. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS, vol. 4585, pp. 726–735. Springer, Heidelberg (2007)
14. Cyran, K.A., Stanczyk, U., Jaroszewicz, L.R.: Subsurface stress monitoring system based on holographic ring-wedge detector and neural network. In: McNulty, G.J. (ed.) Quality, Reliability and Maintenance, pp. 65–68. Professional Engineering Publishing, Bury St Edmunds, London (2002)
15. Doherty, P., Szalas, A.: On the correspondence between approximations and similarity. In: Tsumoto, S., Słowiński, R., Komorowski, J., Grzymała-Busse, J.W. (eds.) RSCTC 2004. LNCS, vol. 3066, pp. 143–152. Springer, Heidelberg (2004)
16. Fares, A., Bouzid, A., Hamdi, M.: Rotation invariance using diffraction pattern sampling in optical pattern recognition. J. of Microwaves and Optoelect. 2(2), 33–39 (2000)
17. Ganotra, D., Joseph, J., Singh, K.: Modified geometry of ring-wedge detector for sampling Fourier transform of fingerprints for classification using neural networks. Proc. SPIE 4829, 407–408 (2003)
18. Ganotra, D., Joseph, J., Singh, K.: Neural network based face recognition by using diffraction pattern sampling with a digital ring-wedge detector. Opt. Comm. 202, 61–68 (2002)
19. George, N., Wang, S.: Neural networks applied to diffraction-pattern sampling. Appl. Opt. 33, 3127–3134 (1994)
20. Gomolinska, A.: A comparative study of some generalized rough approximations. Fundamenta Informaticae 51(1), 103–119 (2002)
21. Grzymala-Busse, J.W.: Data with missing attribute values: Generalization of indiscernibility relation and rule induction. In: Peters, J.F., Skowron, A., Grzymała-Busse, J.W., Kostek, B.z., Świniarski, R.W., Szczuka, M.S. (eds.) Transactions on Rough Sets I. LNCS, vol. 3100, pp. 78–95. Springer, Heidelberg (2004)

22. Grzymala-Busse, J.W.: Rough set strategies to data with missing attribute values. In: Proceedings of the Workshop on Foundations and New Directions in Data Mining, associated with the third IEEE International Conference on Data Mining, Melbourne, FL, USA, November 19– 22, 2003, pp. 56–63 (2003)
23. Jaroszewicz, L.R., Cyran, K.A., Podeszwa, T.: Optimized CGH-based pattern recognizer. Opt. Appl. 30, 317–333 (2000)
24. Jaroszewicz, L.R., Merta, I., Podeszwa, T., Cyran, K.A.: Airplane engine condition monitoring system based on artificial neural network. In: McNulty, G.J. (ed.) Quality, Reliability and Maintenance, pp. 179–182. Professional Engineering Publishing, Bury St Edmunds, London (2002)
25. Jarvinen, J.: Approximations and roughs sets based on tolerances. In: Ziarko, W.P., Yao, Y. (eds.) RSCTC 2000. LNCS (LNAI), vol. 2005, pp. 182–189. Springer, Heidelberg (2001)
26. Kaye, P.H., Barton, J.E., Hirst, E., Clark, J.M.: Simultaneous light scattering and intrinsic fluorescence measurement for the classification of airbone particles. App. Opt. 39(21), 3738–3745 (2000)
27. Kreis, T.: Holographic interferometry: Principles and methods. Akademie Verlag Series in Optical Metrology, vol. 1. Akademie-Verlag (1996)
28. Mait, J.N., Athale, R., van der Gracht, J.: Evolutionary paths in imaging and recent trends. Optics Express 11(18), 2093–2101 (2003)
29. Mrozek, A.: A new method for discovering rules from examples in expert systems. Man-Machine Studies 36, 127–143 (1992)
30. Mrozek, A.: Rough sets in computer implementation of rule-based control of industrial processes. In: Slowinski, R. (ed.) Intelligent decision support. Handbook of applications and advances of the rough sets, pp. 19–31. Kluwer Academic Publishers, Dordrecht (1992)
31. Nebeker, B.M., Hirleman, E.D.: Light scattering by particles and defects on surfaces: semiconductor wafer inspector. Lecture Notes in Physics, vol. 534, pp. 237–257 (2000)
32. Pawlak, Z.: Rough sets: theoretical aspects of reasoning about data. Kluwer Academic, Dordrecht (1991)
33. Piekara, A.H.: New aspects of optics – introduction to quantum electronics and in particular to nonlinear optics and optics of coherent light [in Polish]. PWN, Warsaw (1976)
34. Podeszwa, T., Jaroszewicz, L.R., Cyran, K.A.: Fiberscope based engine condition monitoring system. Proc. SPIE 5124, 299–303 (2003)
35. Skowron, A., Grzymala-Busse, J.W.: From rough set theory to evidence theory. In: Yager, R.R., Ferdizzi, M., Kacprzyk, J. (eds.) Advances in Dempster Shafer theory of evidence, pp. 193–236. Wiley & Sons, NY (1994)
36. Skowron, A., Stepaniuk, J.: Tolerance approximation spaces. Fundamenta Informaticae 27, 245–253 (1996)
37. Slowinski, R., Vanderpooten, D.: A generalized definition of rough approximations based on similarity. IEEE Transaction on Data and Knowledge Engineering 12(2), 331–336 (2000)
38. Slowinski, R., Vanderpooten, D.: Similarity relation as a basis for rough approximations. In: Wang, P.P. (ed.) Advances in machine intelligence and soft computing, pp. 17–33. Bookwrights, Raleigh (1997)
39. Twomey, J.M., Smith, A.E.: Bias and variance of validation methods for function approximation neural networks under conditions of sparse data. IEEE Trans. Sys., Man, and Cyber. 28(3), 417–430 (1998)
40. Ziarko, W.: Variable precision rough set model. J. Comp. Sys. Sci. 40, 39–59 (1993)

On Certain Rough Inclusion Functions

Anna Gomolińska[*]

Białystok University, Department of Mathematics,
Akademicka 2, 15267 Białystok, Poland
anna.gom@math.uwb.edu.pl

Abstract. In this article we further explore the idea which led to the standard rough inclusion function. As a result, two more rough inclusion functions (RIFs in short) are obtained, different from the standard one and from each other. With every RIF we associate a mapping which is in some sense complementary to it. Next, these complementary mappings (co-RIFs) are used to define certain metrics. As it turns out, one of these distance functions is an instance of the Marczewski–Steinhaus metric. While the distance functions may directly be used to measure the degree of dissimilarity of sets of objects, their complementary mappings – also discussed here – are useful in measuring of the degree of mutual similarity of sets.

Keywords: rough inclusion function, rough mereology, distance and similarity between sets.

1 Introduction

Broadly speaking, *rough inclusion functions* (RIFs) are mappings with which one can measure the degree of inclusion of a set in a set. The formal notion of *rough inclusion* was worked out within *rough mereology*, a theory proposed by Polkowski and Skowron [2,3,4]. Rough mereology extends Leśniewski's mereology [5, 6], a formal theory of being-part to the case of being-part-to-degree.

The *standard* RIF is certainly the most famous RIF. Its definition, based on the frequency count, is closely related to the definition of conditional probability. The idea underlying the standard RIF was explored by Łukasiewicz in

[*] This article is an extended version of the paper presented at the International Conference on Rough Sets and Emerging Intelligent Systems Paradigms In Memoriam Zdzisław Pawlak (RSEISP'2007). In comparison to [1], we study properties of the mappings, complementary to the title rough inclusion functions, more intensively. In addition, certain distance functions and their complementary mappings are introduced and investigated. Many thanks to the anonymous referees whose comments helped improve the paper. The research was partly supported by the grant N N516 368334 from Ministry of Science and Higher Education of the Republic of Poland and by an Innovative Economy Operational Programme 2007–2013 (Priority Axis 1. Research and development of new technologies) grant managed by Ministry of Regional Development of the Republic of Poland.

his research on probability of the truth of logical expressions (in particular, implicative formulas) about one hundred years ago [7,8]. Apart from the standard RIF, there are only several functions of such sort described in the literature (see, e.g., [4,9,10]).

Although the notion of RIF is dispensable when approximating sets of objects in line with the classical Pawlak approach [11, 12, 13, 14, 15], it is of particular importance for more general rough-set models. Namely, the concept of RIF is a basic component in Skowron and Stepaniuk's approximation spaces [16, 17, 18, 19,20,21,22] where lower and upper rough approximations of sets of objects are defined by means of RIFs. In the *variable-precision rough-set model* with extensions [23,24,25,26] and the *decision-theoretic rough set model* and its extensions [27,28,29], the standard RIF is taken as an estimator of certain conditional probabilities which, in turn, are used to define variable-precision positive and negative regions of sets of objects. Moreover, starting with a RIF, one can derive a family of *rough membership functions* which was already observed by Pawlak and Skowron in [30]. Also various functions measuring the degree of similarity between sets can be defined by means of RIFs (see, e.g., [4,10,31,32] for the rough-set approach). Last but not the least, a method of knowledge reduction is proposed in [33] which is based, among other things, on the degree of rough inclusion.

In this paper we explore further the idea which led to the standard RIF. The aim is to discover other RIFs which have a similar origin as the standard one. Our investigations are motivated, among other things, by the fact that in spite of well-groundedness, usefulness, and popularity of the standard RIF, some of its properties may seem to be too strong (e.g., Proposition 2a,b). In addition, it would be good to have alternative RIFs at our disposal.

As a result, we have obtained two RIFs more. One of them is new, at least up to the author's knowledge, whereas the remaining one was mentioned in [9]. We investigate properties of the three RIFs with emphasis on the mutual relationships among them. As regards the standard RIF, some of its properties have already been known, but yet a few of them are new. Unlike the standard RIF, the new RIFs do not, at first glance, seem to be very useful to estimate the conditional probability. It turns out however that they are different from, yet definable in terms of the standard RIF. On the other hand, the latter RIF can be derived from the former two.

In the sequel, we introduce mappings complementary to our RIFs, called co-RIFs, and we present their properties. The co-RIFs give rise to certain distance functions which turn out to be metrics on the power set of the universe of objects. The distance functions may directly be used to measure the degree of dissimilarity between sets. It is interesting that one of these metrics is an instance of the Marczewski–Steinhaus metric [34]. Finally, we arrive at mappings complementary to the distance functions. They may, in turn, serve as indices of similarity between sets. It is worthy to note that these similarity indices are known from the literature [35,36,37,38].

The rest of the paper is organized as follows. Section 2 is fully devoted to the standard RIF. In Sect. 3 we recall axioms of rough mereology, a formal theory of being-part-to-degree introduced by Polkowski and Skowron in [2], which provides us with fundamentals of a formal notion of rough inclusion. We also explain what we actually mean by a RIF. In the same section we argue that RIFs indeed realize the formal concept of rough inclusion proposed by Polkowski and Skowron. Some authors [39, 40] (see also [28]) claim rough inclusion measures to fulfil conditions somewhat different from ours. Let us emphasize that our standpoint is that of rough mereology, and its axioms just provide us with a list of postulates to be satisfied by functions measuring the degree of inclusion. In Sect. 4, two alternatives of the standard RIF are derived. In Sect. 5 we consider the co-RIFs corresponding to our three RIFs and we investigate their properties. Certain distance functions and their complementary mappings, induced by the co-RIFs, are discussed in Sect. 6. The last section summarizes the results.

2 The Standard Rough Inclusion Function

The idea underlying the notion of the standard rough inclusion function was explored by Jan Łukasiewicz, a famous Polish logician who, among other things, conducted research on probability of the truth of propositional formulas [7, 8]. The standard RIF is the most popular among functions measuring the degree of inclusion of a set in a set. Let us recall that both the decision-theoretic rough set model [27, 29] and the variable-precision rough set model [23, 24] make use of the standard RIF. It is also commonly used to estimate the *confidence* (or *accuracy*) of decision rules and association rules [10, 41, 42, 43]. Last but not the least, the standard RIF is counted as a function with which one can measure similarity between clusterings [44, 45].

Consider a structure M with a non-empty universe U and a propositional language L interpretable over M. For any formula α and $u \in U$, $u \models \alpha$ reads as 'α is satisfied by u' or 'u satisfies α'. The *extension* of α is defined as the set $||\alpha|| = \{u \in U \mid u \models \alpha\}$. α will be *satisfiable* in M if its extension is non-empty, and *unsatisfiable* otherwise. Morever, α is called *true* in M, $\models \alpha$, if $||\alpha|| = U$. Finally, α entails a formula β, written $\alpha \models \beta$, if and only if every object satisfying α satisfies β as well, i.e., $||\alpha|| \subseteq ||\beta||$. In classical logic, an implicative formula $\alpha \rightarrow \beta$ is true in M if and only if α entails β. Clearly, many interesting formulas are not true in this sense. Since implicative formulas with unsatisfiable predecessors are true, we limit our considerations to satisfiable α. Then, one can assess the degree of truth of $\alpha \rightarrow \beta$ by calculating the probability that an object satisfying α satisfies β as well. Where U is finite, this probability may be estimated by the fraction of objects of $||\alpha||$ which also satisfy β. That is, the degree of truth of $\alpha \rightarrow \beta$ may be defined as $\#(||\alpha|| \cap ||\beta||)/\#||\alpha||$ where $\#||\alpha||$ means the cardinality of $||\alpha||$.

By a straithforward generalization, we arrive at the well-known notion of the standard RIF, commonly used in the rough set theory. It owes its popularity to the clarity of the underlying idea and to the easiness of computation by means of this notion. Since conditional probability may be estimated by the standard RIF, the latter has also been used successfully in the decision-theoretic rough set model [27, 29] (see also [28]) and the variable-precision rough set model and its extensions [23, 24, 25]. Given a non-empty finite set of objects U and its power set $\wp U$, the standard RIF upon U is a mapping $\kappa^{\pounds} : \wp U \times \wp U \mapsto [0,1]$ such that for any $X, Y \subseteq U$,

$$\kappa^{\pounds}(X,Y) \overset{\text{def}}{=} \begin{cases} \frac{\#(X\cap Y)}{\#X} & \text{if } X \neq \emptyset, \\ 1 & \text{otherwise.} \end{cases} \tag{1}$$

To assess the degree of inclusion of a set of objects X in a set of objects Y by means of $\kappa^{\pounds}$, one needs to measure the relative overlap of X with Y. The larger the overlap of two sets, the higher is the degree of inclusion, viz., for any $X, Y, Z \subseteq U$,

$$\#(X \cap Y) \leq \#(X \cap Z) \ \Rightarrow\ \kappa^{\pounds}(X,Y) \leq \kappa^{\pounds}(X,Z).$$

The success of the standard RIF also lies in its mathematical properties. Where $\mathcal{X}$ is a family of sets, we write $\mathrm{Pair}(\mathcal{X})$ to say that elements of $\mathcal{X}$ are pairwise disjoint, i.e., $\forall X, Y \in \mathcal{X}.(X \neq Y \Rightarrow X \cap Y = \emptyset)$. It is assumed that conjunction and disjunction will take the precedence to implication and double implication.

Proposition 1. *For any sets $X, Y, Z \subseteq U$ and any families of sets $\emptyset \neq \mathcal{X}, \mathcal{Y} \subseteq \wp U$, it holds:*

$$\begin{aligned}
&(a)\ \kappa^{\pounds}(X,Y) = 1 \ \Leftrightarrow\ X \subseteq Y,\\
&(b)\ Y \subseteq Z \ \Rightarrow\ \kappa^{\pounds}(X,Y) \leq \kappa^{\pounds}(X,Z),\\
&(c)\ Z \subseteq Y \subseteq X \ \Rightarrow\ \kappa^{\pounds}(X,Z) \leq \kappa^{\pounds}(Y,Z),\\
&(d)\ \kappa^{\pounds}(X, \bigcup\mathcal{Y}) \leq \sum_{Y\in\mathcal{Y}} \kappa^{\pounds}(X,Y),\\
&(e)\ X \neq \emptyset \ \&\ \mathrm{Pair}(\mathcal{Y}) \ \Rightarrow\ \kappa^{\pounds}(X, \bigcup\mathcal{Y}) = \sum_{Y\in\mathcal{Y}} \kappa^{\pounds}(X,Y),\\
&(f)\ \kappa^{\pounds}(\bigcup\mathcal{X}, Y) \leq \sum_{X\in\mathcal{X}} \kappa^{\pounds}(X,Y)\cdot\kappa^{\pounds}(\bigcup\mathcal{X}, X),\\
&(g)\ \mathrm{Pair}(\mathcal{X}) \ \Rightarrow\ \kappa^{\pounds}(\bigcup\mathcal{X}, Y) = \sum_{X\in\mathcal{X}} \kappa^{\pounds}(X,Y)\cdot\kappa^{\pounds}(\bigcup\mathcal{X}, X).
\end{aligned}$$

Proof. We prove (f) only. Consider any $Y \subseteq U$ and any non-empty family $\mathcal{X} \subseteq \wp U$. First suppose that $\bigcup\mathcal{X} = \emptyset$, i.e., $\mathcal{X} = \{\emptyset\}$. The property obviously holds since $\kappa^{\pounds}(\bigcup\mathcal{X}, Y) = 1$ and $\kappa^{\pounds}(\bigcup\mathcal{X}, \emptyset)\cdot\kappa^{\pounds}(\emptyset, Y) = 1\cdot 1 = 1$. Now let $\bigcup\mathcal{X}$ be non-empty. In such a case, $\kappa^{\pounds}(\bigcup\mathcal{X}, Y) = \#(\bigcup\mathcal{X}\cap Y)/\#\bigcup\mathcal{X} = \#\bigcup\{X \cap Y \mid X \in \mathcal{X}\}/\#\bigcup\mathcal{X} \leq \sum\{\#(X\cap Y) \mid X \in \mathcal{X}\}/\#\bigcup\mathcal{X} = \sum\{\#(X\cap Y)/\#\bigcup\mathcal{X} \mid X \in \mathcal{X}\}$. Observe that if some element X of $\mathcal{X}$ is empty, then $\#(X \cap Y)/\#\bigcup\mathcal{X} = 0$. On the other hand, $\kappa^{\pounds}(X,Y) \cdot \kappa^{\pounds}(\bigcup\mathcal{X}, X) = 1 \cdot (\#X/\#\bigcup\mathcal{X}) = 1 \cdot 0 = 0$ as well. For every non-empty element X of $\mathcal{X}$, we have $\#(X \cap Y)/\#\bigcup\mathcal{X} = (\#(X\cap Y)/\#X)\cdot(\#X/\#\bigcup\mathcal{X}) = \kappa^{\pounds}(X,Y)\cdot\kappa^{\pounds}(\bigcup\mathcal{X}, X)$ as required. Summing up, $\kappa^{\pounds}(\bigcup\mathcal{X}, Y) \leq \sum_{X\in\mathcal{X}} \kappa^{\pounds}(X,Y)\cdot\kappa^{\pounds}(\bigcup\mathcal{X}, X)$. □

Some comments can be useful here. (a) says that the standard RIF yields 1 if and only if the first argument is included in the second one. Property (b) expresses monotonicity of $\kappa^{£}$ in the second variable, whereas (c) states some weak form of co-monotonicity of the standard RIF in the first variable. It follows from (d) that for any covering of a set of objects, say Z, the sum of the degrees of inclusion of a set X in the sets constituting the covering is at least as high as the degree of inclusion of X in Z. The non-strict inequality in (d) may be strenghtened to $=$ for non-empty X and coverings consisting of pairwise disjoint sets as stated by (e). Due to (f), for any covering of a set of objects, say Z, the degree of inclusion of Z in a set Y is not higher than a weighted sum of the degrees of inclusion of sets constituting the covering in Y where the weights are the degrees of inclusion of Z in the members of the covering of Z. In virtue of (g), the inequality may be strenghtened to $=$ if elements of the covering are pairwise disjoint. Let us observe that (g) is in some sense a counterpart of the total probability theorem.

The following conclusions can be drawn from the facts above.

Proposition 2. *For any $X, Y, Z, W \subseteq U$ $(X \neq \emptyset)$ and a family $\mathcal{Y}$ of pairwise disjoint sets of objects such that $\bigcup \mathcal{Y} = U$, we have:*

$$(a)\ \sum_{Y \in \mathcal{Y}} \kappa^{£}(X, Y) = 1,$$

$$(b)\ \kappa^{£}(X, Y) = 0 \ \Leftrightarrow\ X \cap Y = \emptyset,$$

$$(c)\ \kappa^{£}(X, \emptyset) = 0,$$

$$(d)\ X \cap Y = \emptyset \ \Rightarrow\ \kappa^{£}(X, Z - Y) = \kappa^{£}(X, Z \cup Y) = \kappa^{£}(X, Z),$$

$$(e)\ Z \cap W = \emptyset \ \Rightarrow\ \kappa^{£}(Y \cup Z, W) \leq \kappa^{£}(Y, W) \leq \kappa^{£}(Y - Z, W),$$

$$(f)\ Z \subseteq W \ \Rightarrow\ \kappa^{£}(Y - Z, W) \leq \kappa^{£}(Y, W) \leq \kappa^{£}(Y \cup Z, W).$$

Proof. We show (d) only. To this end, consider any sets of objects X, Y where $X \neq \emptyset$ and $X \cap Y = \emptyset$. Immediately (d1) $\kappa^{£}(X, Y) = 0$ by (b). Hence, for any $Z \subseteq U$, $\kappa^{£}(X, Z) = \kappa^{£}(X, (Z \cap Y) \cup (Z - Y)) = \kappa^{£}(X, Z \cap Y) + \kappa^{£}(X, Z - Y) \leq \kappa^{£}(X, Y) + \kappa^{£}(X, Z - Y) = \kappa^{£}(X, Z - Y)$ in virtue of Proposition 1b,e. In the sequel, $\kappa^{£}(X, Z \cup Y) \leq \kappa^{£}(X, Z) + \kappa^{£}(X, Y) = \kappa^{£}(X, Z)$ due to (d1) and Proposition 1d. The remaining inequalities are consequences of Proposition 1b. □

Let us note a few remarks. (a) states that the degrees of inclusion of a non-empty set of objects X in pairwise disjoint sets will sum up to 1 when these sets, taken together, cover the universe. In virtue of (b), the degree of inclusion of a non-empty set in an arbitrary set of objects equals to 0 just in the case the both sets are disjoint. (b) obviously implies (c). The latter property says that the degree of inclusion of a non-empty set in $\emptyset$ is equal to 0. Thanks to (d), removing (resp., adding) objects, not being members of a non-empty set X, from (to) a set Z does not influence the degree of inclusion of X in Z. As follows from (e), adding (resp., removing) objects, not belonging to a set W, to (from) a set Y does not increase (decrease) the degree of inclusion of Y in W. Finally, removing (resp., adding) members of a set of objects W from (to) a set Y does not increase (decrease) the degree of inclusion of Y in W due to (f).

Example 1. Given $U = \{0, \ldots, 9\}$, $X = \{0, \ldots, 3\}$, $Y = \{0, \ldots, 3, 8\}$, and $Z = \{2, \ldots, 6\}$. Note that $X \cap Z = Y \cap Z = \{2, 3\}$. Thus, $\kappa^{£}(X, Z) = 1/2$ and $\kappa^{£}(Z, X) = 2/5$ which means that the standard RIF is not symmetric. Moreover, $\kappa^{£}(Y, Z) = 2/5 < 1/2$. Thus, $X \subseteq Y$ may not imply $\kappa^{£}(X, Z) \leq \kappa^{£}(Y, Z)$, i.e., $\kappa^{£}$ is not monotone in the first variable.

3 Rough Mereology: A Formal Framework for Rough Inclusion

The notion of the standard RIF was generalized and formalized by Polkowski and Skowron within rough mereology, a theory of the notion of being-part-to-degree [2, 3, 4]. The starting point is a pair of formal theories introduced by Leśniewski [5, 6], viz., mereology and ontology where the former theory extends the latter one. Mereology is a theory of the notion of being-part, whereas ontology is a theory of names and plays the role of set theory. Leśniewski's mereology is also known as a theory of collective sets as opposite to ontology being a theory of distributive sets. In this section we only recall a very small part of rough mereology, pivotal for the notion of rough inclusion. We somewhat change the original notation (e.g., 'el' to 'ing', 'μ_t' to 'ing_t'), yet trying to keep with the underlying ideas.

In ontology, built upon the classical predicate logic with identity, two basic semantical categories are distinguished: the category of non-empty names[1] and the category of propositions. We use x, y, z, with subscripts if needed, as name variables and we denote the set of all such variables by Var. The only primitive notion of ontology is the copula 'is', denoted by ε and characterized by the axiom

$$(L0)\;\; x\varepsilon y \leftrightarrow (\exists z.z\varepsilon x \wedge \forall z, z'.(z\varepsilon x \wedge z'\varepsilon x \rightarrow z\varepsilon z') \wedge \forall z.(z\varepsilon x \rightarrow z\varepsilon y)) \qquad (2)$$

where '$x\varepsilon y$' is read as 'x is y'. The first two conjuncts on the right-hand side say that x ranges over non-empty, individual names only. The third conjunct says that each of x's is y as well. In particular, the intended meaning of '$x\varepsilon x$' is simply that x ranges over individual names.

Mereology is built upon ontology and introduces a name-forming functor pt where '$x\varepsilon\mathrm{pt}(y)$' reads as '$x$ is a *part* of y'. The functor pt is described by the following axioms:

$$\begin{aligned}
&(L1)\; x\varepsilon\mathrm{pt}(y) \rightarrow x\varepsilon x \wedge y\varepsilon y,\\
&(L2)\; x\varepsilon\mathrm{pt}(y) \wedge y\varepsilon\mathrm{pt}(z) \rightarrow x\varepsilon\mathrm{pt}(z),\\
&(L3)\; \neg(x\varepsilon\mathrm{pt}(x)).
\end{aligned}$$

(L1) stipulates that both x and y range over individual names. According to (L2) and (L3), being-part is transitive and irreflexive, respectively. The reflexive counterpart of pt is the notion of *being-ingredient*, ing, given by

$$x\varepsilon\mathrm{ing}(y) \overset{\mathrm{def}}{\leftrightarrow} x\varepsilon\mathrm{pt}(y) \vee x = y. \qquad (3)$$

[1] Empty names are denied by Leśniewski on philosophical grounds.

One can see that

$$
\begin{aligned}
&(L1')\ x\varepsilon\mathrm{ing}(y) \rightarrow x\varepsilon x \wedge y\varepsilon y,\\
&(L2')\ x\varepsilon\mathrm{ing}(y) \wedge y\varepsilon\mathrm{ing}(z) \rightarrow x\varepsilon\mathrm{ing}(z),\\
&(L3')\ x\varepsilon\mathrm{ing}(x),\\
&(L4')\ x\varepsilon\mathrm{ing}(y) \wedge y\varepsilon\mathrm{ing}(x) \rightarrow x = y.
\end{aligned}
$$

Axioms (L1'), (L2') are counterparts of (L1), (L2), respectively. (L3'), (L4') postulate reflexivity and antisymmetry of ing, respectively. It is worth noting that one can start with ing characterized by (L1')–(L4') and define pt by

$$x\varepsilon\mathrm{pt}(y) \overset{\mathrm{def}}{\leftrightarrow} x\varepsilon\mathrm{ing}(y) \wedge x \neq y. \tag{4}$$

Polkowski and Skowron's rough mereology extends Leśniewski's mereology by a family of name-forming functors ing_t. These functors, constituting a formal counterpart of the notion of being-ingredient-to-degree, are described by the following axioms, for any name variables x, y, z and $s, t \in [0, 1]$:

$$
\begin{aligned}
&(PS1)\ \exists t.x\varepsilon\mathrm{ing}_t(y) \rightarrow x\varepsilon x \wedge y\varepsilon y,\\
&(PS2)\ x\varepsilon\mathrm{ing}_1(y) \leftrightarrow x\varepsilon\mathrm{ing}(y),\\
&(PS3)\ x\varepsilon\mathrm{ing}_1(y) \rightarrow \forall z.(z\varepsilon\mathrm{ing}_t(x) \rightarrow z\varepsilon\mathrm{ing}_t(y)),\\
&(PS4)\ x = y \wedge x\varepsilon\mathrm{ing}_t(z) \rightarrow y\varepsilon\mathrm{ing}_t(z),\\
&(PS5)\ x\varepsilon\mathrm{ing}_t(y) \wedge s \leq t \rightarrow x\varepsilon\mathrm{ing}_s(y).
\end{aligned}
$$

The expression '$x\varepsilon\mathrm{ing}_t(y)$' reads as '$x$ is an ingredient of y to degree t'. The axiom (PS1) claims x, y to range over individual names. According to (PS2), being an ingredient to degree 1 is equivalent with being an ingredient. (PS3) states a weak form of transitivity of the graded ingredienthood. (PS4) says that '=' is congruencial with respect to being-ingredient-to-degree. As postulated by (PS5), ing_t is, in fact, a formalization of the notion of being an ingredient to degree at least t. Furthermore, being-part-to-degree may be defined as a special case of the graded ingredienthood, viz.,

$$x\varepsilon\mathrm{pt}_t(y) \overset{\mathrm{def}}{\leftrightarrow} x\varepsilon\mathrm{ing}_t(y) \wedge x \neq y. \tag{5}$$

The axioms (PS1)–(PS5) are minimal conditions to be fulfilled by the formal concept of graded ingredienthood[2]. According to the standard interpretation, being an ingredient (part) is understood as being included (included in the proper sense). In the same vein, the graded ingredienthood may be interpreted as a graded inclusion, called *rough inclusion* in line with Polkowski and Skowron.

Now we describe a model for the part of rough mereology presented above, simplifying the picture as much as possible. Consider a non-empty set of objects U and a structure $M = (\wp U, \subseteq, \kappa)$ where the set of all subsets of U, $\wp U$, serves

[2] For instance, nothing has been said about the property of being external yet. For this and other concepts of rough mereology see, e.g., [4].

as the universe of M, $\subseteq$ is the usual inclusion relation on $\wp U$, and κ is a mapping $\kappa : \wp U \times \wp U \mapsto [0,1]$ satisfying the conditions $\mathrm{rif}_1, \mathrm{rif}_2$ below:

$$\mathrm{rif}_1(\kappa) \overset{\text{def}}{\Leftrightarrow} \forall X, Y \subseteq U.(\kappa(X,Y) = 1 \ \Leftrightarrow\ X \subseteq Y),$$
$$\mathrm{rif}_2(\kappa) \overset{\text{def}}{\Leftrightarrow} \forall X, Y, Z \subseteq U.(Y \subseteq Z \ \Rightarrow\ \kappa(X,Y) \leq \kappa(X,Z)).$$

According to rif_1, κ is a generalization of $\subseteq$. Moreover, κ achieves the greatest value (equal to 1) only for such pairs of sets that the second element of a pair contains the first element. The condition rif_2 postulates κ to be monotone in the second variable. We call any mapping κ as above a *rough inclusion function* (RIF) over U. For simplicity, the reference to U will be dropped if no confusion results. Observe that having assumed rif_1, the second condition is equivalent to rif_2^* given by

$$\mathrm{rif}_2^*(\kappa) \overset{\text{def}}{\Leftrightarrow} \forall X, Y, Z \subseteq U.(\kappa(Y,Z) = 1 \ \Rightarrow\ \kappa(X,Y) \leq \kappa(X,Z)).$$

Subsets of U are viewed as concepts, and RIFs are intended as functions measuring the degrees of inclusion of concepts in concepts. It is worth noting that any RIF over U is a fuzzy set on $\wp U \times \wp U$ or, in other words, a fuzzy binary relation on $\wp U$ (see [46] and more recent, ample literature on fuzzy set theory). Clearly, RIFs may satisfy various additional postulates as well. Examples of such postulates are:

$$\begin{aligned}
\mathrm{rif}_3(\kappa) &\overset{\text{def}}{\Leftrightarrow} \forall \emptyset \neq X \subseteq U.\kappa(X,\emptyset) = 0,\\
\mathrm{rif}_4(\kappa) &\overset{\text{def}}{\Leftrightarrow} \forall X, Y \subseteq U.(\kappa(X,Y) = 0 \ \Rightarrow\ X \cap Y = \emptyset),\\
\mathrm{rif}_4^{-1}(\kappa) &\overset{\text{def}}{\Leftrightarrow} \forall \emptyset \neq X \subseteq U.\forall Y \subseteq U.(X \cap Y = \emptyset \ \Rightarrow\ \kappa(X,Y) = 0),\\
\mathrm{rif}_5(\kappa) &\overset{\text{def}}{\Leftrightarrow} \forall \emptyset \neq X \subseteq U.\forall Y \subseteq U.(\kappa(X,Y) = 0 \ \Leftrightarrow\ X \cap Y = \emptyset),\\
\mathrm{rif}_6(\kappa) &\overset{\text{def}}{\Leftrightarrow} \forall \emptyset \neq X \subseteq U.\forall Y \subseteq U.\kappa(X,Y) + \kappa(X, U - Y) = 1,\\
\mathrm{rif}_7(\kappa) &\overset{\text{def}}{\Leftrightarrow} \forall X, Y, Z \subseteq U.(Z \subseteq Y \subseteq X \ \Rightarrow\ \kappa(X,Z) \leq \kappa(Y,Z)).
\end{aligned}$$

As follows from Propositions 1 and 2, the standard RIF satisfies all the conditions above. Moreover, for any RIF κ, $\mathrm{rif}_1(\kappa)$ and $\mathrm{rif}_6(\kappa)$ imply $\mathrm{rif}_5(\kappa)$; $\mathrm{rif}_5(\kappa)$ is equivalent to the conjunction of $\mathrm{rif}_4(\kappa)$ and $\mathrm{rif}_4^{-1}(\kappa)$; and $\mathrm{rif}_4^{-1}(\kappa)$ implies $\mathrm{rif}_3(\kappa)$. It is worth mentioning that some authors stipulate functions measuring the degree of inclusion to satisfy rif_2, rif_7, and the 'if' part of rif_1 [39, 40].

Names and name-forming functors are interpreted in M by means of a mapping I as follows. Every name is interpreted as a non-empty set of concepts, i.e., subsets of U, and individual names are interpreted as singletons. For any singleton $Y = \{X\}$ where $X \subseteq U$, let

$$e(Y) \overset{\text{def}}{=} X. \tag{6}$$

The identity symbol is interpreted as the identity relation on $\wp U$ (the same symbol '=' is used in both cases for simplicity). The copula ε is interpreted as a binary relation $\varepsilon_I \subseteq \wp(\wp U) \times \wp(\wp U)$ such that for any $X, Y \subseteq \wp U$,

$$X \varepsilon_I Y \overset{\text{def}}{\Leftrightarrow} \#X = 1 \;\&\; X \subseteq Y. \tag{7}$$

Observe that $X \subseteq Y$ above may equivalently be written as $e(X) \in Y$. In the sequel, the name-forming functors ing, pt, ing_t, and pt_t ($t \in [0,1]$) are interpreted as mappings $\text{ing}_I, \text{pt}_I, \text{ing}_{t,I}, \text{pt}_{t,I} : \wp U \mapsto \wp(\wp U)$ such that for any $X \subseteq U$,

$$\begin{aligned}
\text{ing}_I(X) &\overset{\text{def}}{=} \wp X, \\
\text{pt}_I(X) &\overset{\text{def}}{=} \wp X - \{X\}, \\
\text{ing}_{t,I}(X) &\overset{\text{def}}{=} \{Y \subseteq U \mid \kappa(Y,X) \geq t\}, \\
\text{pt}_{t,I}(X) &\overset{\text{def}}{=} \{Y \subseteq U \mid \kappa(Y,X) \geq t \;\&\; Y \neq X\},
\end{aligned} \tag{8}$$

thus, e.g., ing is interpreted as the power-set operator. The pair $M_I = (M, I)$ is an interpretation of the language of the part of rough mereology considered here.

In the next step, we assign non-empty sets of concepts to name variables. Given an interpretation M_I, any such variable assignment $v : \text{Var} \mapsto \wp(\wp U)$ may be extended to a term assignment v_I as follows. For any $x \in \text{Var}$, $t \in [0,1]$, and $f \in \{\text{ing}, \text{pt}, \text{ing}_t, \text{pt}_t\}$,

$$\begin{aligned}
v_I(x) &\overset{\text{def}}{=} v(x), \\
v_I(f(x)) &\overset{\text{def}}{=} \begin{cases} f_I(e(v(x))) & \text{if } \#v(x) = 1, \\ \text{undefined} & \text{otherwise.} \end{cases}
\end{aligned} \tag{9}$$

Finally, we can define satisfiability of formulas by variable assignments in M_I. For any formula α and any variable assignment v, '$M_I, v \models \alpha$' reads as 'α is satisfied by v in M_I'. Along the standard lines, α will be *true in* M_I, $M_I \models \alpha$, if α is satisfied by every variable assignment in M_I. The relation of satisfiability of formulas is defined as follows, for any formulas α, β, any name variables x, y, any degree variable t, and $f \in \{\text{ing}, \text{pt}, \text{ing}_t, \text{pt}_t\}$:

$$\begin{aligned}
M_I, v \models x = y &\overset{\text{def}}{\Leftrightarrow} v_I(x) = v_I(y), \\
M_I, v \models x \varepsilon y &\overset{\text{def}}{\Leftrightarrow} v_I(x) \varepsilon_I v_I(y), \\
M_I, v \models x \varepsilon f(y) &\overset{\text{def}}{\Leftrightarrow} v_I(x) \varepsilon_I v_I(f(y)), \\
M_I, v \models \alpha \wedge \beta &\overset{\text{def}}{\Leftrightarrow} M_I, v \models \alpha \;\&\; M_I, v \models \beta, \\
M_I, v \models \neg\alpha &\overset{\text{def}}{\Leftrightarrow} M_I, v \not\models \alpha, \\
M_I, v \models \forall x.\alpha &\overset{\text{def}}{\Leftrightarrow} M_I, w \models \alpha \text{ for any } w \text{ different from } v \text{ at most for } x, \\
M_I, v \models \forall t.\alpha &\overset{\text{def}}{\Leftrightarrow} \text{ for every } t \in [0,1],\; M_I, v \models \alpha.
\end{aligned} \tag{10}$$

The remaining cases can easily be obtained from those above. Let us observe that the first three conditions may be simplified to the following ones:

$$M_I, v \models x = y \Leftrightarrow v(x) = v(y),$$

$$\begin{aligned}
M_I, v \models x\varepsilon y &\Leftrightarrow (\#v(x) = 1 \;\&\; v(x) \subseteq v(y)) \\
&\Leftrightarrow \exists X \subseteq U.(v(x) = \{X\} \;\&\; X \in v(y)), \\
M_I, v \models x\varepsilon\mathrm{ing}(y) &\Leftrightarrow (\#v(x) = \#v(y) = 1 \;\&\; e(v(x)) \subseteq e(v(y))) \\
&\Leftrightarrow \exists X, Y \subseteq U.(v(x) = \{X\} \;\&\; v(y) = \{Y\} \;\&\; X \subseteq Y), \\
M_I, v \models x\varepsilon\mathrm{pt}(y) &\Leftrightarrow (\#v(x) = \#v(y) = 1 \;\&\; e(v(x)) \subset e(v(y))) \\
&\Leftrightarrow \exists X, Y \subseteq U.(v(x) = \{X\} \;\&\; v(y) = \{Y\} \;\&\; X \subset Y), \\
M_I, v \models x\varepsilon\mathrm{ing}_t(y) &\Leftrightarrow (\#v(x) = \#v(y) = 1 \;\&\; \kappa(e(v(x)), e(v(y))) \geq t) \\
&\Leftrightarrow \exists X, Y \subseteq U.(v(x) = \{X\} \;\&\; v(y) = \{Y\} \;\&\; \kappa(X, Y) \geq t), \\
M_I, v \models x\varepsilon\mathrm{pt}_t(y) &\Leftrightarrow (\#v(x) = \#v(y) = 1 \;\&\; v(x) \neq v(y) \\
&\quad \& \; \kappa(e(v(x)), e(v(y))) \geq t) \\
&\Leftrightarrow \exists X, Y \subseteq U.(v(x) = \{X\} \;\&\; v(y) = \{Y\} \;\&\; X \neq Y \\
&\quad \& \; \kappa(X, Y) \geq t). \qquad (11)
\end{aligned}$$

By a straightforward inspection one can check that M_I is a model of the considered part of rough mereology, i.e., all axioms are true in M_I. By way of example, we only show that (PS3) is true in M_I, i.e., for any name variables x, y, any $t \in [0, 1]$, and any variable assignment v,

$$M_I, v \models x\varepsilon\mathrm{ing}_1(y) \rightarrow \forall z.(z\varepsilon\mathrm{ing}_t(x) \rightarrow z\varepsilon\mathrm{ing}_t(y)). \qquad (12)$$

To this end, assume $M_I, v \models x\varepsilon\mathrm{ing}_1(y)$ first. Hence, (a) $\#v(x) = \#v(y) = 1$ and $\kappa(e(v(x)), e(v(y))) \geq 1$ by (11). The latter is equivalent with (b) $e(v(x)) \subseteq e(v(y))$ due to $\mathrm{rif}_1(\kappa)$. Next consider any variable assignment w, different from v at most for z. As a consequence, (c) $w(x) = v(x)$ and $w(y) = v(y)$. In the sequel assume $M_I, w \models z\varepsilon\mathrm{ing}_t(x)$. Hence, (d) $\#w(z) = 1$ and (e) $\kappa(e(w(z)), e(w(x))) \geq t$ by (11). It holds that (f) $\kappa(e(w(z)), e(w(x))) \leq \kappa(e(w(z)), e(w(y)))$ by (b), (c), and $\mathrm{rif}_2(\kappa)$. From the latter and (e) we obtain (g) $\kappa(e(w(z)), e(w(y))) \geq t$. Hence, $M_I, w \models z\varepsilon\mathrm{ing}_t(y)$ in virtue of (a), (c), (d), and (11).

4 In Search of New RIFs

According to rough mereology, rough inclusion is a generalization of the set-theoretical inclusion of sets. While keeping with this idea, we try to obtain RIFs different from the standard one. Let U be a non-empty finite set of objects. Observe that for any $X, Y \subseteq U$, the following formulas are equivalent:

$$\begin{aligned}
&(i)\;\; X \subseteq Y, \\
&(ii)\;\; X \cap Y = X, \\
&(iii)\; X \cup Y = Y, \\
&(iv)\;\; (U - X) \cup Y = U, \\
&(v)\;\; X - Y = \emptyset. \qquad (13)
\end{aligned}$$

The equivalence of the first two statements gave rise to the standard RIF. Now we explore $(i) \Leftrightarrow (iii)$ and $(i) \Leftrightarrow (iv)$. In the case of (iii), '$\supseteq$' always holds true.

Conversely, '$\subseteq$' always takes place in (iv). The remaining inclusions may or may not hold, so we may introduce degrees of inclusion. Thus, let us define mappings $\kappa_1, \kappa_2 : \wp U \times \wp U \mapsto [0,1]$ such that for any $X, Y \subseteq U$,

$$\kappa_1(X,Y) \overset{\text{def}}{=} \begin{cases} \frac{\#Y}{\#(X \cup Y)} & \text{if } X \cup Y \neq \emptyset, \\ 1 & \text{otherwise,} \end{cases}$$

$$\kappa_2(X,Y) \overset{\text{def}}{=} \frac{\#((U-X) \cup Y)}{\#U}. \tag{14}$$

It is worth noting that κ_2 was mentioned in [9]. Now we show that both κ_1, κ_2 are RIFs different from the standard one and from each other.

Proposition 3. *Each of* κ_i *(*$i = 1, 2$*) is a RIF upon* U*, i.e.,* $\mathrm{rif}_1(\kappa_i)$ *and* $\mathrm{rif}_2(\kappa_i)$ *hold.*

Proof. We only prove the property for $i = 1$. Let X, Y, Z be any sets of objects. To show $\mathrm{rif}_1(\kappa_1)$, we only examine the non-trivial case where $X, Y \neq \emptyset$. Then, $\kappa_1(X,Y) = 1$ if and only if $\#Y = \#(X \cup Y)$ if and only if $Y = X \cup Y$ if and only if $X \subseteq Y$. In the case of rif_2 assume that (a1) $Y \subseteq Z$. First suppose that $X = \emptyset$. If Z is empty as well, then $Y = \emptyset$. In result, $\kappa_1(X,Y) = 1 \leq 1 = \kappa_1(X,Z)$. Conversely, if Z is non-empty, then $\kappa_1(X,Z) = \#Z/\#Z = 1 \geq \kappa_1(X,Y)$. Now assume that $X \neq \emptyset$. Then $X \cup Y, X \cup Z \neq \emptyset$. Moreover, $Z = Y \cup (Z - Y)$ and $Y \cap (Z - Y) = \emptyset$ by (a1). As a consequence, (a2) $\#Z = \#Y + \#(Z - Y)$. Additionally (a3) $\#(X \cup Z) \leq \#(X \cup Y) + \#(Z - Y)$ and (a4) $\#Y \leq \#(X \cup Y)$. Hence, $\kappa_1(X,Y) = \#Y/\#(X \cup Y) \leq (\#Y + \#(Z-Y))/(\#(X \cup Y) + \#(Z-Y)) \leq (\#Y + \#(Z - Y))/\#(X \cup Y \cup (Z - Y)) = \#Z/\#(X \cup Z) = \kappa_1(X,Z)$ by (a2)–(a4). $\square$

Example 2. Consider $U = \{0, \ldots, 9\}$ and its subsets $X = \{0, \ldots, 4\}$, $Y = \{2, \ldots, 6\}$. Notice that $X \cap Y = \{2, 3, 4\}$, $X \cup Y = \{0, \ldots, 6\}$, and $(U - X) \cup Y = \{2, \ldots, 9\}$. Hence, $\kappa^{£}(X,Y) = 3/5$, $\kappa_1(X,Y) = 5/7$, and $\kappa_2(X,Y) = 4/5$, i.e., $\kappa^{£}$, κ_1, and κ_2 are different RIFs.

Proposition 4. *For any* $X, Y \subseteq U$*, we have:*

(a) $X \neq \emptyset \Rightarrow (\kappa_1(X,Y) = 0 \Leftrightarrow Y = \emptyset)$,
(b) $\kappa_2(X,Y) = 0 \Leftrightarrow X = U \;\&\; Y = \emptyset$,
(c) $\mathrm{rif}_4(\kappa_1) \;\&\; \mathrm{rif}_4(\kappa_2)$,
(d) $\kappa^{£}(X,Y) \leq \kappa_1(X,Y) \leq \kappa_2(X,Y)$,
(e) $\kappa_1(X,Y) = \kappa^{£}(X \cup Y, Y)$,
(f) $\kappa_2(X,Y) = \kappa^{£}(U, (U - X) \cup Y) = \kappa^{£}(U, U - X) + \kappa^{£}(U, X \cap Y)$,
(g) $\kappa^{£}(X,Y) = \kappa^{£}(X, X \cap Y) = \kappa_1(X, X \cap Y) = \kappa_1(X - Y, X \cap Y)$,
(h) $X \cup Y = U \Rightarrow \kappa_1(X,Y) = \kappa_2(X,Y)$.

Proof. By way of illustration we show (d) and (h). To this end, consider any sets of objects X, Y. In case (d), if X is empty, then $(U - X) \cup Y = U$. Hence by the definitions, $\kappa^{\pounds}(X,Y) = \kappa_1(X,Y) = \kappa_2(X,Y) = 1$. Now suppose that $X \neq \emptyset$. Obviously (d1) $\#(X \cap Y) \leq \#X$ and (d2) $\#Y \leq \#(X \cup Y)$. Since $X \cup Y = X \cup (Y - X)$ and $X \cap (Y - X) = \emptyset$, (d3) $\#(X \cup Y) = \#X + \#(Y - X)$. Similarly, it follows from $Y = (X \cap Y) \cup (Y - X)$ and $(X \cap Y) \cap (Y - X) = \emptyset$ that (d4) $\#Y = \#(X \cap Y) + \#(Y - X)$. Observe also that $(U - X) \cup Y = ((U - X) - Y) \cup Y = (U - (X \cup Y)) \cup Y$ and $(U - (X \cup Y)) \cap Y = \emptyset$. Hence, (d5) $\#((U - X) \cup Y) = \#(U - (X \cup Y)) + \#Y$. In the sequel, $\kappa^{\pounds}(X,Y) = \#(X \cap Y)/\#X \leq (\#(X \cap Y) + \#(Y - X))/(\#X + \#(Y - X)) = \#Y/\#(X \cup Y) = \kappa_1(X,Y) \leq (\#(U - (X \cup Y)) + \#Y)/(\#(U - (X \cup Y)) + \#(X \cup Y)) = \#((U - X) \cup Y)/\#U = \kappa_2(X,Y)$ by (d1)–(d5) and the definitions of the RIFs.

For (h) assume that $X \cup Y = U$. Then $Y - X = U - X$, and $\kappa_1(X,Y) = \#Y/\#U = \#((Y - X) \cup Y)/\#U = \#((U - X) \cup Y)/\#U = \kappa_2(X,Y)$ as required. $\square$

Let us briefly comment upon the properties. According to (a), if X is non-empty, then the emptiness of Y will be both sufficient[3] and necessary to have $\kappa_1(X,Y) = 0$. Property (b) states that κ_2 yields 0 solely for $(U, \emptyset)$. Due to (c), $\kappa_i(X,Y) = 0$ $(i = 1, 2)$ implies the emptiness of the overlap of X, Y. Property (d) says that the degrees of inclusion yielded by κ_2 are at least as high as those given by κ_1, and the degrees of inclusion provided by κ_1 are not lower than those estimated by means of the standard RIF. (e) and (f) provide us with characterizations of κ_1 and κ_2 in terms of $\kappa^{\pounds}$, respectively. On the other hand, the standard RIF may be defined by means of κ_1 in virtue of (g). Finally, (h) states that κ_1, κ_2 are equal on the set of all pairs (X,Y) such that X, Y cover the universe.

5 Mappings Complementary to RIFs

Now we define mappings which are in some sense complementary to the RIFs considered. We also investigate properties of these functions and give one more characterization of the standard RIF. Namely, with every mapping $f : \wp U \times \wp U \mapsto [0,1]$ one can associate a complementary mapping $\bar{f} : \wp U \times \wp U \mapsto [0,1]$ defined by

$$\bar{f}(X,Y) \overset{\text{def}}{=} 1 - f(X,Y) \tag{15}$$

for any sets $X, Y \subseteq U$. Clearly, f is complementary to $\bar{f}$. In particular, we obtain

$$\bar{\kappa}^{\pounds}(X,Y) = \begin{cases} \frac{\#(X-Y)}{\#X} & \text{if } X \neq \emptyset, \\ 0 & \text{otherwise,} \end{cases}$$

$$\bar{\kappa}_1(X,Y) = \begin{cases} \frac{\#(X-Y)}{\#(X \cup Y)} & \text{if } X \cup Y \neq \emptyset, \\ 0 & \text{otherwise,} \end{cases}$$

$$\bar{\kappa}_2(X,Y) = \frac{\#(X - Y)}{\#U}. \tag{16}$$

[3] Compare the optional postulate $\text{rif}_3(\kappa)$.

For the sake of simplicity, $\bar{\kappa}$ where κ is a RIF will be referred to as a co-RIF. Observe that each of the co-RIFs measures the difference between its first and second arguments, i.e., the equivalence $(i) \Leftrightarrow (v)$ (cf. (13)) is explored here.

It is worthy to note that for any $X, Y \subseteq U$,

$$\bar{\kappa}^{\pounds}(X,Y) = \kappa^{\pounds}(X, U-Y). \tag{17}$$

However, the same is not true of $\bar{\kappa}_i$ for $i = 1, 2$. Indeed, $\kappa_1(X, U-Y) = \#(U-Y)/\#(X \cup (U-Y))$ if $X \cup (U-Y) \neq \emptyset$, and $\kappa_2(X, U-Y) = \#(U-(X \cap Y))/\#U$, so the counterparts of (17) do not hold in general.

Example 3. Let U and X, Y be as in Example 2, i.e., $U = \{0, \ldots, 9\}$, $X = \{0, \ldots, 4\}$, and $Y = \{2, \ldots, 6\}$. It is easy to see that $\kappa_1(X, U-Y) = 5/8$ and $\kappa_2(X, U-Y) = 7/10$, whereas $\bar{\kappa}_1(X,Y) = 2/7$ and $\bar{\kappa}_2(X,Y) = 1/5$.

We can characterize the standard RIF in terms of κ_i $(i = 1, 2)$ and their co-RIFs as follows:

Proposition 5. *For any sets of objects X, Y where $X \neq \emptyset$,*

$$\kappa^{\pounds}(X,Y) = \frac{\bar{\kappa}_1(X, U-Y)}{\kappa_1(U-Y, X)} = \frac{\bar{\kappa}_2(X, U-Y)}{\kappa_2(U, X)}.$$

Proof. Consider any set of objects Y and any non-empty set of objects X. Hence, $X \cup (U-Y) \neq \emptyset$ as well. Moreover, $\kappa_1(U-Y, X), \kappa_2(U, X) > 0$. Then $\bar{\kappa}_1(X, U-Y) = \#(X-(U-Y))/\#(X \cup (U-Y)) = \#(X \cap Y)/\#(X \cup (U-Y)) = (\#(X \cap Y)/\#X) \cdot (\#X/\#(X \cup (U-Y))) = \kappa^{\pounds}(X,Y) \cdot \kappa_1(U-Y, X)$ by the definitions of $\kappa^{\pounds}$, κ_1, and $\bar{\kappa}_1$. Hence, $\kappa^{\pounds}(X,Y) = \bar{\kappa}_1(X, U-Y)/\kappa_1(U-Y, X)$ as required. Similarly, $\bar{\kappa}_2(X, U-Y) = \#(X-(U-Y))/\#U = \#(X \cap Y)/\#U = (\#(X \cap Y)/\#X) \cdot (\#X/\#U) = \kappa^{\pounds}(X,Y) \cdot \kappa_2(U, X)$ by the definitions of $\kappa^{\pounds}$, κ_2, and $\bar{\kappa}_2$. Immediately $\kappa^{\pounds}(X,Y) = \bar{\kappa}_2(X, U-Y)/\kappa_2(U, X)$ which ends the proof. $\square$

Henceforth the symmetric difference of sets X, Y will be denoted by $X \div Y$. We can prove the following properties of co-RIFs:

Proposition 6. *For any $X, Y, Z \subseteq U$, an arbitrary RIF κ, and $i = 1, 2$,*

$$\begin{aligned}
&(a)\ \bar{\kappa}(X,Y) = 0 \ \Leftrightarrow\ X \subseteq Y,\\
&(b)\ Y \subseteq Z \ \Rightarrow\ \bar{\kappa}(X,Z) \subseteq \bar{\kappa}(X,Y),\\
&(c)\ \bar{\kappa}_2(X,Y) \leq \bar{\kappa}_1(X,Y) \leq \bar{\kappa}^{\pounds}(X,Y),\\
&(d)\ \bar{\kappa}_i(X,Y) + \bar{\kappa}_i(Y,Z) \geq \bar{\kappa}_i(X,Z),\\
&(e)\ 0 \leq \bar{\kappa}_i(X,Y) + \bar{\kappa}_i(Y,X) \leq 1,\\
&(f)\ (X = \emptyset \ \&\ Y \neq \emptyset)\ \ \mathit{or}\ \ (X \neq \emptyset \ \&\ Y = \emptyset) \ \Rightarrow\\
&\qquad \bar{\kappa}^{\pounds}(X,Y) + \bar{\kappa}^{\pounds}(Y,X) = \bar{\kappa}_1(X,Y) + \bar{\kappa}_1(Y,X) = 1.
\end{aligned}$$

Proof. We only prove (d) for $i = 1$, and (e). To this end, consider any sets of objects X, Y, Z. In case (d), if $X = \emptyset$, then $\bar{\kappa}_1(X,Z) = 0$ in virtue of (a). Hence,

(d) obviously holds. Now suppose that $X \neq \emptyset$. If $Y = \emptyset$, then $\bar{\kappa}_1(X,Y) = 1$. On the other hand, if $Y \neq \emptyset$ and $Z = \emptyset$, then $\bar{\kappa}_1(Y,Z) = 1$. In both cases $\bar{\kappa}_1(X,Y) + \bar{\kappa}_1(Y,Z) \geq 1 \geq \bar{\kappa}_1(X,Z)$.

Finally, assume that $X, Y, Z \neq \emptyset$. Let $m = \#(X \cup Y \cup Z)$, $m_0 = \#(X - (Y \cup Z))$, $m_1 = \#(Y - (X \cup Z))$, $m_2 = \#((X \cap Y) - Z)$, $m_3 = \#((X \cap Z) - Y)$, and $m_4 = \#(Z - (X \cup Y))$. Observe that $\#(X - Y) = m_0 + m_3$, $\#(X - Z) = m_0 + m_2$, $\#(Y - Z) = m_1 + m_2$, $\#(X \cup Y) = m - m_4$, $\#(X \cup Z) = m - m_1$, and $\#(Y \cup Z) = m - m_0$. Hence, $\bar{\kappa}_1(X,Y) = \#(X - Y)/\#(X \cup Y) = (m_0 + m_3)/(m - m_4)$. On the same grounds, $\bar{\kappa}_1(Y,Z) = (m_1 + m_2)/(m - m_0)$ and $\bar{\kappa}_1(X,Z) = (m_0 + m_2)/(m - m_1)$. It is easy to see that

$$\frac{m_0 + m_3}{m - m_4} + \frac{m_1 + m_2}{m - m_0} \geq \frac{m_0 + m_3}{m} + \frac{m_1 + m_2}{m} \geq \frac{m_0 + m_1 + m_2}{m} \geq \frac{m_0 + m_2}{m - m_1}$$

which ends the proof of (d).

The first inequality of (e) is obvious, so we only show the second one. For $i = 1$ assume that $X \cup Y \neq \emptyset$ since the case $X = Y = \emptyset$ is trivial. Thus, $\bar{\kappa}_1(X,Y) + \bar{\kappa}_1(Y,X) = (\#(X - Y)/\#(X \cup Y)) + (\#(Y - X)/\#(X \cup Y)) = \#(X \div Y)/\#(X \cup Y) \leq 1$ because $X \div Y \subseteq X \cup Y$. The property just proved implies the second inequality for $i = 2$ due to (c). $\square$

According to (a), every co-RIF will yield 0 exactly in the case the first argument is included in the second one. As a consequence, (*) $\bar{\kappa}(X,X) = 0$ for every set of objects X. That is, $\bar{\kappa}$ may serve as a (non-symmetric) distance function. (b) states that co-RIFs are co-monotone in the second variable. (c) provides us with a comparison of our three co-RIFs. Properties (e), (f) will prove their usefulness in the next section. (d) expresses the *triangle inequality* condition for $\bar{\kappa}_i$ $(i = 1, 2)$.

Let us note that the triangle inequality does not hold for $\bar{\kappa}^{\pounds}$ in general.

Example 4. Consider sets of objects X, Y, Z such that $X - Z, Z - X \neq \emptyset$ and $Y = X \cup Z$. We show that

$$\bar{\kappa}^{\pounds}(X,Y) + \bar{\kappa}^{\pounds}(Y,Z) < \bar{\kappa}^{\pounds}(X,Z).$$

By the assumptions each of X, Y, Z is non-empty and $Y - Z = X - Z$. Next, $\#X < \#Y$ since $X \subset Y$. Moreover, $\bar{\kappa}^{\pounds}(X,Y) = 0$ in virtue of (a). As a consequence,

$$\bar{\kappa}^{\pounds}(X,Y) + \bar{\kappa}^{\pounds}(Y,Z) = \frac{\#(Y - Z)}{\#Y} < \frac{\#(Y - Z)}{\#X} = \frac{\#(X - Z)}{\#X} = \bar{\kappa}^{\pounds}(X,Z)$$

as expected.

Additionally, it can be that $\bar{\kappa}^{\pounds}(X,Y) + \bar{\kappa}^{\pounds}(Y,X) > 1$. Indeed, if $X, Y \neq \emptyset$ and $X \cap Y = \emptyset$, then $\Sigma = \bar{\kappa}^{\pounds}(X,Y) + \bar{\kappa}^{\pounds}(Y,X) = (\#X/\#X) + (\#Y/\#Y) = 1 + 1 = 2$. Nevertheless, 2 is the greatest value taken by Σ.

6 RIFs and Their Complementary Mappings vs. Similarity and Distance between Sets

In this section we use the three co-RIFs to define certain normalized distance functions with which one can measure (dis)similarity between sets. Namely, let $\delta^{£}, \delta_i : \wp U \times \wp U \mapsto [0,1]$ $(i = 1, 2)$ be mappings such that for any $X, Y \subseteq U$,

$$\begin{aligned}\delta^{£}(X,Y) &\overset{\text{def}}{=} \frac{1}{2}\left(\bar{\kappa}^{£}(X,Y) + \bar{\kappa}^{£}(Y,X)\right), \\ \delta_i(X,Y) &\overset{\text{def}}{=} \bar{\kappa}_i(X,Y) + \bar{\kappa}_i(Y,X). \end{aligned} \tag{18}$$

It is easy to see that

$$\begin{aligned}\delta^{£}(X,Y) &= \begin{cases} \frac{1}{2}\left(\frac{\#(X-Y)}{\#X} + \frac{\#(Y-X)}{\#Y}\right) & \text{if } X, Y \neq \emptyset, \\ 0 & \text{if } X, Y = \emptyset, \\ \frac{1}{2} & \text{in the remaining cases,} \end{cases} \\ \delta_1(X,Y) &= \begin{cases} \frac{\#(X \div Y)}{\#(X \cup Y)} & \text{if } X \cup Y \neq \emptyset, \\ 0 & \text{otherwise,} \end{cases} \\ \delta_2(X,Y) &= \frac{\#(X \div Y)}{\#U}. \end{aligned} \tag{19}$$

It is worth mentioning that δ_1 is an instance of the Marczewski–Steinhaus metric [34]. As we shall see, the remaining two functions are metrics on $\wp U$ as well. Namely, we can prove the following:

Proposition 7. *For any sets $X, Y, Z \subseteq U$ and $\delta \in \{\delta^{£}, \delta_1, \delta_2\}$,*

$$\begin{aligned} &(a)\ \delta(X,Y) = 0 \Leftrightarrow X = Y, \\ &(b)\ \delta(X,Y) = \delta(Y,X), \\ &(c)\ \delta(X,Y) + \delta(Y,Z) \geq \delta(X,Z), \\ &(d)\ \max\{\delta^{£}(X,Y), \delta_2(X,Y)\} \leq \delta_1(X,Y) \leq 2\delta^{£}(X,Y). \end{aligned}$$

Proof. Property (a) is an easy consequence of Proposition 6a. (b) directly follows from the definitions of $\delta^{£}$, δ_1, and δ_2. Property (c) for δ_1, δ_2 can easily be obtained from Proposition 6d. Now we show (c) for $\delta^{£}$. To this end, consider any sets of objects X, Y, Z.

If both X, Z are empty, then $\delta^{£}(X,Z) = 0$ in virtue of (a), and (c) follows immediately. Next, if $X, Y = \emptyset$ and $Z \neq \emptyset$, or $X, Y \neq \emptyset$ and $Z = \emptyset$, then $\delta^{£}(X,Z) = \delta^{£}(Y,Z) = 1/2$ by (19). In consequence, (c) is fulfilled regardless of the value $\delta^{£}(X,Y)$. In the same vein, if $X = \emptyset$ and $Y, Z \neq \emptyset$, or $X \neq \emptyset$ and $Y, Z = \emptyset$, then $\delta^{£}(X,Y) = \delta^{£}(X,Z) = 1/2$. Here is (c) satisfied regardless of $\delta^{£}(Y,Z)$. In the sequel, if $X, Z \neq \emptyset$ and $Y = \emptyset$, then $\delta^{£}(X,Y) = \delta^{£}(Y,Z) = 1/2$. Hence, $\delta^{£}(X,Y) + \delta^{£}(Y,Z) = 1 \geq \delta^{£}(X,Z)$ for any value of $\delta^{£}$ at (X,Z).

Finally we prove (c) for $X, Y, Z \neq \emptyset$. Let m and m_i ($i = 0, \ldots, 4$) be as earlier. Additionally, let $m_5 = \#((Y \cap Z) - X)$. Notice that

$$\begin{aligned}
2\delta^{\pounds}(X,Y) &= \frac{\#(X-Y)}{\#X} + \frac{\#(Y-X)}{\#Y} \\
&= \frac{m_0+m_3}{m-(m_1+m_4+m_5)} + \frac{m_1+m_5}{m-(m_0+m_3+m_4)}, \\
2\delta^{\pounds}(Y,Z) &= \frac{\#(Y-Z)}{\#Y} + \frac{\#(Z-Y)}{\#Z} \\
&= \frac{m_1+m_2}{m-(m_0+m_3+m_4)} + \frac{m_3+m_4}{m-(m_0+m_1+m_2)}, \\
2\delta^{\pounds}(X,Z) &= \frac{\#(X-Z)}{\#X} + \frac{\#(Z-X)}{\#Z} \\
&= \frac{m_0+m_2}{m-(m_1+m_4+m_5)} + \frac{m_4+m_5}{m-(m_0+m_1+m_2)}.
\end{aligned}$$

Hence we obtain

$$\begin{aligned}
&2(\delta^{\pounds}(X,Y) + \delta^{\pounds}(Y,Z) - \delta^{\pounds}(X,Z)) = \\
&\quad \frac{(m_0+m_3)-(m_0+m_2)}{m-(m_1+m_4+m_5)} + \frac{(m_1+m_5)+(m_1+m_2)}{m-(m_0+m_3+m_4)} + \frac{(m_3+m_4)-(m_4+m_5)}{m-(m_0+m_1+m_2)} \\
&\quad \geq \frac{m_3-m_2}{m} + \frac{2m_1+m_2+m_5}{m} + \frac{m_3-m_5}{m} = \frac{2(m_1+m_3)}{m} \geq 0.
\end{aligned}$$

In result, $\delta^{\pounds}(X,Y) + \delta^{\pounds}(Y,Z) \geq \delta^{\pounds}(X,Z)$ as needed.

As regards (d), we only prove that (*) $\delta^{\pounds}(X,Y) \leq \delta_1(X,Y)$ for any $X, Y \subseteq U$. The rest easily follows from Proposition 6c. Consider any sets of objects X, Y. If at least one of X, Y is empty, (*) will directly hold by the definitions of $\delta^{\pounds}, \delta_1$. For the remaining case observe that

$$\frac{\#(X \cap Y)}{\#(X \cup Y)} \leq \min\left\{\frac{\#(X \cap Y)}{\#X}, \frac{\#(X \cap Y)}{\#Y}\right\}$$

since $\max\{\#X, \#Y\} \leq \#(X \cup Y)$. Hence, we obtain in the sequel:

$$\begin{aligned}
\max\left\{1 - \frac{\#(X \cap Y)}{\#X}, 1 - \frac{\#(X \cap Y)}{\#Y}\right\} &\leq 1 - \frac{\#(X \cap Y)}{\#(X \cup Y)}, \\
\max\left\{\frac{\#(X-Y)}{\#X}, \frac{\#(Y-X)}{\#Y}\right\} &\leq \frac{\#(X \div Y)}{\#(X \cup Y)}, \\
\frac{\#(X-Y)}{\#X} + \frac{\#(Y-X)}{\#Y} &\leq 2\frac{\#(X \div Y)}{\#(X \cup Y)}, \\
\frac{1}{2}\left(\frac{\#(X-Y)}{\#X} + \frac{\#(Y-X)}{\#Y}\right) &\leq \frac{\#(X \div Y)}{\#(X \cup Y)}.
\end{aligned}$$

From the latter we derive (*) by the definitions of $\delta^{\pounds}, \delta_1$. □

Summing up, $\delta^{\pounds}$ and δ_i ($i = 1, 2$) are metrics on $\wp U$ due to (a)–(c), and they may be used to measure the distance between sets. According to (d), the double distance between sets X, Y, estimated by means of $\delta^{\pounds}$, will be not smaller than the distance between X, Y yielded by δ_1. In turn, the distance measured by the latter metric will be greater than or equal to the distance given by each of $\delta^{\pounds}, \delta_2$. In view of the fact that $\bar{\kappa}_i$, underlying δ_i, satisfy the triangle inequality (see Proposition 6d), it is not very surprizing that δ_i are metrics. The really unexpected result is that $\delta^{\pounds}$ fulfils the triangle inequality as well.

The distance between two sets may be interpreted as the degree of their dissimilarity. Thus, $\delta^{\pounds}$ and δ_i may serve as measures (indices) of dissimilarity of sets. On the other hand, mappings which are complementary in the sense of (15) to $\delta^{\pounds}$ and δ_i, $\bar{\delta}^{\pounds}$ and $\bar{\delta}_i$ ($i = 1, 2$), respectively, may be used as similarity measures (see, e.g., [44] for a discussion of various indices used to measure the degree of similarity between clusterings). Let us note that for any $X, Y \subseteq U$, the following dependencies hold:

$$\begin{aligned} \bar{\delta}^{\pounds}(X,Y) &= \frac{1}{2}\left(\kappa^{\pounds}(X,Y) + \kappa^{\pounds}(Y,X)\right), \\ \bar{\delta}_i(X,Y) &= \kappa_i(X,Y) + \kappa_i(Y,X) - 1. \end{aligned} \tag{20}$$

More precisely,

$$\begin{aligned} \bar{\delta}^{\pounds}(X,Y) &= \begin{cases} \frac{\#(X\cap Y)}{2}\left(\frac{1}{\#X} + \frac{1}{\#Y}\right) & \text{if } X, Y \neq \emptyset, \\ 1 & \text{if } X, Y = \emptyset, \\ \frac{1}{2} & \text{in the remaining cases,} \end{cases} \\ \bar{\delta}_1(X,Y) &= \begin{cases} \frac{\#(X\cap Y)}{\#(X\cup Y)} & \text{if } X \cup Y \neq \emptyset, \\ 1 & \text{otherwise,} \end{cases} \\ \bar{\delta}_2(X,Y) &= \frac{\#((U - (X\cup Y)) \cup (X \cap Y))}{\#U}. \end{aligned} \tag{21}$$

Thus, starting with the standard RIF and two other RIFs of a similar origin, we have finally arrived at similarity measures known from the literature [35, 36, 37, 38]. More precisely, $\bar{\delta}^{\pounds}$ is the function proposed by Kulczyński to estimate biotopical similarity [36]. The similarity index $\bar{\delta}_1$, complementary to the Marczewski–Steinhaus metric δ_1, is attributed to Jaccard [35]. The function $\bar{\delta}_2$ was introduced (at least) twice, viz., by Sokal and Michener [38] and by Rand [37].

Let us note the following observations:

Proposition 8. *For any sets of objects X, Y and $\delta \in \{\delta^{\pounds}, \delta_1, \delta_2\}$, we have that:*

(*a*) $\bar{\delta}(X,Y) = 1 \Leftrightarrow X = Y$,

(*b*) $\bar{\delta}(X,Y) = \bar{\delta}(Y,X)$,

(*c*) $\bar{\delta}^{\pounds}(X,Y) = 0 \Leftrightarrow X \cap Y = \emptyset \;\&\; X, Y \neq \emptyset$,

(*d*) $\bar{\delta}_1(X,Y) = 0 \Leftrightarrow X \cap Y = \emptyset \;\&\; X \cup Y \neq \emptyset$,

(*e*) $\bar{\delta}_2(X,Y) = 0 \Leftrightarrow X \cap Y = \emptyset \ \& \ X \cup Y = U$,
(*f*) $2\bar{\delta}^{\pounds}(X,Y) - 1 \leq \bar{\delta}_1(X,Y) \leq \min\{\bar{\delta}^{\pounds}(X,Y), \bar{\delta}_2(X,Y)\}$.

The proof is easy and, hence, omitted. However, some remarks may be useful. (a) states that every set is similar to itself to the highest degree 1. According to (b), similarity is assumed to be symmetric here. Properties (c)–(e) describe conditions characterizing the lowest degree of similarity between sets. A comparison of the three similarity indices is provided by (f).

An example, illustrating a possible application of the Marczewski–Steinhaus metric to estimate differences between biotopes, can be found in [34]. In that example, two real forests from Lower Silesia (Poland) are considered. We slightly modify the example and we extend it to the other distance measures investigated.

Example 5. As the universe we take a collection of tree species $U = \{\mathtt{a},\mathtt{b},\mathtt{h},\mathtt{l},\mathtt{o},\mathtt{p},\mathtt{r},\mathtt{s}\}$ where `a` stands for ‘alder’, `b` – ‘birch’, `h` – ‘hazel’, `l` – ‘larch’, `o` – ‘oak’, `p` – ‘pine’, `r` – ‘rowan’, and `s` – ‘spruce’. Consider two forests represented by the collections A, B of the tree species which occur in those forests where $A = \{\mathtt{a},\mathtt{b},\mathtt{h},\mathtt{p},\mathtt{r}\}$ and $B = \{\mathtt{b},\mathtt{o},\mathtt{p},\mathtt{s}\}$. First we compute the degrees of inclusion of A in B, and vice-versa. Next we measure the biotopical differences between A and B using $\delta^{\pounds}$ and δ_i for $i = 1,2$. Finally we estimate the degrees of biotopical similarity of the forests investigated. It is easy to see that $\kappa^{\pounds}(A,B) = 2/5$, $\kappa^{\pounds}(B,A) = 1/2$, $\kappa_1(A,B) = 4/7$, $\kappa_1(B,A) = 5/7$, $\kappa_2(A,B) = 5/8$, and $\kappa_2(B,A) = 3/4$. Hence,

$$\delta^{\pounds}(A,B) = \frac{1}{2}\left(\frac{3}{5} + \frac{1}{2}\right) = \frac{11}{20},$$

$\delta_1(A,B) = 5/7$, and $\delta_2(A,B) = 5/8$. As expected, the distance functions $\delta^{\pounds}, \delta_1, \delta_2$ (and so the corresponding similarity measures $\bar{\delta}^{\pounds}, \bar{\delta}_1, \bar{\delta}_2$) may give us different values when measuring the distance (resp., similarity) between A and B. Due to Proposition 7d, this distance is the greatest (equal to 5/7) when measured by δ_1. Conversely, $\bar{\delta}_1$ yields the least degree of similarity, equal to 2/7. Therefore, these measures seem to be particularly attractive to cautious reasoners. For those who accept a higher risk, both $\delta^{\pounds}, \delta_2$ (and similarly, $\bar{\delta}^{\pounds}, \bar{\delta}_2$) are reasonable alternatives too. Accidentially, $\delta^{\pounds}$ gives the least distance, equal to 11/20, and its complementary mapping $\bar{\delta}^{\pounds}$ yields the greatest degree of similarity, equal to 9/20. In this particular case, values provided by δ_2 and $\bar{\delta}_2$, 5/8 and 3/8, respectively, are in between. Clearly, the choice of the most appropriate distance function (or similarity measure) may also depend on factors other than the level of risk.

7 Summary

In this article, an attempt was made to discover RIFs different from the standard one, yet having a similar origin. First we overviewed the notion of the standard RIF, $\kappa^{\pounds}$. In the next step, a general framework for discussion of RIFs and their properties was recalled. As a result, a minimal set of postulates specifying a RIF was derived. Also several optional conditions were proposed. Then we defined two RIFs, κ_1 and κ_2, which turned out to be different from the standard one. The latter RIF was mentioned in [9], yet the former one seems to be new. We examined properties of these RIFs with a special stress laid on the relationship to the standard RIF.

In the sequel, we introduced functions complementary to RIFs (co-RIFs) which resulted in a new characterization of the standard RIF in terms of the remaining two RIFs and their complementary mappings. We examined properties of each of the three co-RIFs: $\bar{\kappa}^{\pounds}$, $\bar{\kappa}_1$, and $\bar{\kappa}_2$. We easily found out that they might serve as distance functions. However, only the latter two functions proved to satisfy the triangle inequality.

In the next step, the co-RIFs were used to define certain distance functions, $\delta^{\pounds}$, δ_1, and δ_2, which turned out to be metrics on the power set of the set of all objects considered. δ_1 has already been known in the literature [34]. From the distance functions mentioned above we finally derived their complementary mappings, $\bar{\delta}^{\pounds}$, $\bar{\delta}_1$, and $\bar{\delta}_2$, serving as similarity measures. As turned out, they were discovered many years ago [35,36,37,38]. In this way, starting with an idea which led to the standard RIF and going through intermediate stages (co-RIFs and certain metrics based on them), we finally arrived at similarity indices known in machine learning, relational learning, and statistical learning, to name a few areas of application.

References

1. Gomolińska, A.: On three closely related rough inclusion functions. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 142–151. Springer, Heidelberg (2007)
2. Polkowski, L., Skowron, A.: Rough mereology. In: Raś, Z.W., Zemankova, M. (eds.) ISMIS 1994. LNCS (LNAI), vol. 869, pp. 85–94. Springer, Heidelberg (1994)
3. Polkowski, L., Skowron, A.: Rough mereology: A new paradigm for approximate reasoning. Int. J. Approximated Reasoning 15, 333–365 (1996)
4. Polkowski, L., Skowron, A.: Rough mereological calculi of granules: A rough set approach to computation. Computational Intelligence 17, 472–492 (2001)
5. Leśniewski, S.: Foundations of the General Set Theory 1 (in Polish). Works of the Polish Scientific Circle, Moscow, vol. 2 (1916); Also In: [6], pp. 128–173
6. Surma, S.J., Srzednicki, J.T., Barnett, J.D. (eds.): Stanisław Leśniewski Collected Works. Kluwer/Polish Scientific Publ., Dordrecht/Warsaw (1992)
7. Borkowski, L. (ed.): Jan Łukasiewicz – Selected Works. North Holland/Polish Scientific Publ., Amsterdam/Warsaw (1970)
8. Łukasiewicz, J.: Die logischen Grundlagen der Wahrscheinlichkeitsrechnung, Cracow (1913); In: [7], pp. 16-63 (English translation)
9. Drwal, G., Mrózek, A.: System RClass – software implementation of a rough classifier. In: Kłopotek, M.A., Michalewicz, M., Raś, Z.W. (eds.) Proc. 7th Int. Symp. Intelligent Information Systems (IIS 1998), Malbork, Poland, June 1998, pp. 392–395 (1998)

10. Stepaniuk, J.: Knowledge discovery by application of rough set models. In: Polkowski, L., Tsumoto, S., Lin, T.Y. (eds.) Rough Set Methods and Applications: New Developments in Knowledge Discovery in Information Systems, pp. 137–233. Physica, Heidelberg (2001)
11. Pawlak, Z.: Rough sets. Int. J. Computer and Information Sciences 11, 341–356 (1982)
12. Pawlak, Z.: Information Systems. Theoretical Foundations (in Polish). Wydawnictwo Naukowo-Techniczne, Warsaw (1983)
13. Pawlak, Z.: Rough Sets. Theoretical Aspects of Reasoning About Data. Kluwer, Dordrecht (1991)
14. Pawlak, Z.: Rough set elements. In: Polkowski, L., Skowron, A. (eds.) Rough Sets in Knowledge Discovery, vol. 1, pp. 10–30. Physica, Heidelberg (1998)
15. Pawlak, Z.: A treatise on rough sets. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets IV. LNCS, vol. 3700, pp. 1–17. Springer, Heidelberg (2005)
16. Bazan, J.G., Skowron, A., Swiniarski, R.: Rough sets and vague concept approximation: From sample approximation to adaptive learning. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets V. LNCS, vol. 4100, pp. 39–63. Springer, Heidelberg (2006)
17. Peters, J.F.: Approximation spaces for hierarchical intelligent behavioral system models. In: Dunin-Kęplicz, B., Jankowski, A., Skowron, A., Szczuka, M. (eds.) Monitoring, Security, and Rescue Techniques in Multiagent Systems, pp. 13–30. Springer, Heidelberg (2005)
18. Peters, J.F., Skowron, A., Stepaniuk, J.: Nearness of objects: Extension of approximation space model. Fundamenta Informaticae 79, 497–512 (2007)
19. Skowron, A., Stepaniuk, J.: Generalized approximation spaces. In: Lin, T.Y., Wildberger, A.M. (eds.) Soft Computing. Simulation Councils, pp. 18–21. Simulation Councils, San Diego (1995)
20. Skowron, A., Stepaniuk, J.: Tolerance approximation spaces. Fundamenta Informaticae 27, 245–253 (1996)
21. Skowron, A., Stepaniuk, J., Peters, J.F., Swiniarski, R.: Calculi of approximation spaces. Fundamenta Informaticae 72, 363–378 (2006)
22. Skowron, A., Swiniarski, R., Synak, P.: Approximation spaces and information granulation. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets III. LNCS, vol. 3400, pp. 175–189. Springer, Heidelberg (2005)
23. Ziarko, W.: Variable precision rough set model. J. Computer and System Sciences 46, 39–59 (1993)
24. Ziarko, W.: Probabilistic decision tables in the variable precision rough set model. Computational Intelligence 17, 593–603 (2001)
25. Ziarko, W.: Probabilistic rough sets. In: Ślęzak, D., Wang, G., Szczuka, M.S., Düntsch, I., Yao, Y. (eds.) RSFDGrC 2005. LNCS, vol. 3641, pp. 283–293. Springer, Heidelberg (2005)
26. Ziarko, W.: Stochastic approach to rough set theory. In: Greco, S., Hata, Y., Hirano, S., Inuiguchi, M., Miyamoto, S., Nguyen, H.S., Słowiński, R. (eds.) RSCTC 2006. LNCS, vol. 4259, pp. 38–48. Springer, Heidelberg (2006)
27. Yao, Y.Y.: Decision-theoretic rough set models. In: Yao, J., Lingras, P., Wu, W.-Z., Szczuka, M.S., Cercone, N.J., Ślęzak, D. (eds.) RSKT 2007. LNCS, vol. 4481, pp. 1–12. Springer, Heidelberg (2007)
28. Yao, Y.Y.: Probabilistic rough set approximations. Int. J. of Approximate Reasoning (in press, 2007), doi:10.1016/j.ijar.2007.05.019
29. Yao, Y.Y., Wong, S.K.M.: A decision theoretic framework for approximating concepts. Int. J. of Man–Machine Studies 37, 793–809 (1992)

30. Pawlak, Z., Skowron, A.: Rough membership functions. In: Fedrizzi, M., Kacprzyk, J., Yager, R.R. (eds.) Advances in the Dempster–Shafer Theory of Evidence, pp. 251–271. John Wiley & Sons, Chichester (1994)
31. Gomolińska, A.: Possible rough ingredients of concepts in approximation spaces. Fundamenta Informaticae 72, 139–154 (2006)
32. Nguyen, H.S., Skowron, A., Stepaniuk, J.: Granular computing: A rough set approach. Computational Intelligence 17, 514–544 (2001)
33. Zhang, M., Xu, L.D., Zhang, W.X., Li, H.Z.: A rough set approach to knowledge reduction based on inclusion degree and evidence reasoning theory. Expert Systems 20, 298–304 (2003)
34. Marczewski, E., Steinhaus, H.: On a certain distance of sets and the corresponding distance of functions. Colloquium Mathematicum 6, 319–327 (1958)
35. Jaccard, P.: Nouvelles recherches sur la distribution florale. Bull. de la Société Vaudoise des Sciences Naturelles 44, 223–270 (1908)
36. Kulczyński, S.: Die Pflanzenassociationen der Pieninen. Bull. Internat. Acad. Polon. Sci. Lett., Sci. Math. et Naturelles, serie B, suppl. II 2, 57–203 (1927)
37. Rand, W.: Objective criteria for the evaluation of clustering methods. J. of the American Statistical Association 66, 846–850 (1971)
38. Sokal, R.R., Michener, C.D.: A statistical method for evaluating systematic relationships. University of Kansas Science Bulletin 38, 1409–1438 (1958)
39. Xu, Z.B., Liang, J.Y., Dang, C.Y., Chin, K.S.: Inclusion degree: A perspective on measures for rough set data analysis. Information Sciences 141, 227–236 (2002)
40. Zhang, W.X., Leung, Y.: Theory of including degrees and its applications to uncertainty inference. In: Proc. of 1996 Asian Fuzzy System Symposium, pp. 496–501 (1996)
41. An, A., Cercone, N.: Rule quality measures for rule induction systems: Description and evaluation. Computational Intelligence 17, 409–424 (2001)
42. Kryszkiewicz, M.: Fast discovery of representative association rules. In: Polkowski, L., Skowron, A. (eds.) RSCTC 1998. LNCS (LNAI), vol. 1424, pp. 214–221. Springer, Heidelberg (1998)
43. Tsumoto, S.: Modelling medical diagnostic rules based on rough sets. In: Polkowski, L., Skowron, A. (eds.) RSCTC 1998. LNCS (LNAI), vol. 1424, pp. 475–482. Springer, Heidelberg (1998)
44. Albatineh, A.N., Niewiadomska-Bugaj, M., Mihalko, D.: On similarity indices and correction for chance agreement. J. of Classification 23, 301–313 (2006)
45. Wallace, D.L.: A method for comparing two hierarchical clusterings: Comment. J. of the American Statistical Association 78, 569–576 (1983)
46. Zadeh, L.A.: Fuzzy sets. Information and Control 8, 338–353 (1965)

Automatic Rhythm Retrieval from Musical Files

Bożena Kostek, Jarosław Wójcik, and Piotr Szczuko

Gdańsk University of Technology, Multimedia Systems Department
Narutowicza 11/12, 80-952 Gdańsk, Poland
{bozenka,szczuko}@sound.eti.pg.gda.pl,
jaroslaw.wojcik@intesk.pl

Abstract. This paper presents a comparison of the effectiveness of two computational intelligence approaches applied to the task of retrieving rhythmic structure from musical files. The method proposed by the authors of this paper generates rhythmic levels first, and then uses these levels to compose rhythmic hypotheses. Three phases: creating periods, creating simplified hypotheses and creating full hypotheses are examined within this study. All experiments are conducted on a database of national anthems. Decision systems such as Artificial Neural Networks and Rough Sets are employed to search the metric structure of musical files. This was based on examining physical attributes of sound that are important in determining the placement of a particular sound in the accented location of a musical piece. The results of the experiments show that both decision systems award note duration as the most significant parameter in automatic searching for metric structure of rhythm from musical files. Also, a brief description of the application realizing automatic rhythm accompaniment is presented.

Keywords: Rhythm Retrieval, Metric Rhythm, Music Information Retrieval, Artificial Neural Networks, Rough Sets.

1 Introduction

The aim of this article is to present a comparative study of the effectiveness of two computational intelligence approaches applied to the task of retrieving rhythmic structure from musical files. Existing metric rhythm research usually focuses on retrieving low rhythmic levels – they go down to the level of a measure. Typically those methods are sufficient to emulate human perception of a local rhythm. According to McAuley & Semple [14] trained musicians perceive more levels, though. High-level perception is required from drum players, thus computational approach needs to retrieve a so-called hypermetric structure of a piece. If it reaches high rhythmic levels such as phrases, sentences and periods, then automatic drum accompaniment applications can be developed.

Rhythm retrieval research is a broad field and, among other issues, involves the quantization process of the beginnings and lengths of notes, the extraction of rhythm events from audio recordings, and the search for meter of compositions.

Rhythm is an element of a piece determining musical style, which may be valuable in retrieval. The rhythmic structure together with patterns retrieved carry information about the genre of a piece. Content-based methods of music retrieval are nowadays developed by researchers from the multimedia retrieval and computational intelligence domain. The most common classes of rhythm retrieval models are: rule-based, multiple-agents, multiple-oscillators and probabilistic. The rhythm retrieval methods can be classified within the context of what type of actions they take, i.e. whether they quantize musical data, or find the tempo of a piece (e.g. van Belle [2]), time signatures, positions of barlines, a metric structure or an entire hypermetric hierarchy. Rhythm finding systems very often rank the hypotheses of rhythm, basing on the sound salience function. Since scientists differ in opinions on the aspect of salience, the Authors carried out special experiments to solve the salience problem. A number of research studies are based on the theory published by Lerdahl & Jackendoff [13], who claim that such physical attributes of sounds as pitch (frequency), duration and velocity (amplitude) influence the rhythmical salience of sounds. Another approach, proposed by Rosenthal [19], ranks higher the hypotheses in which long sounds are placed in accented positions. In Dixon's [4] multiple-agent approach, two salience functions are proposed, combining duration, pitch and velocity. The first, is a linear combination of physical attributes, Dixon calls it an *additive function.* The other one is a *multiplicative function.* Dahl [3] notices that drummers play accented strokes with higher amplitude than unaccented ones. Parncutt, in his book [15], claims that lower sounds fall on the beat. In the review of Parncutt's book, Huron [5] notices that the high salience of low sounds is "neither an experimentally determined fact nor an established principle in musical practice". A duration-based hypothesis predominated in rhythm-related works, however this approach seemed to be based on intuition only. The experimental confirmation of this thesis – based on the Data Mining (DM) association rules and Artificial Neural Networks (ANNs) – can be found in former works by the Authors of this paper [6], [7], [8] and also in the doctoral thesis of Wojcik [27]. The experiments employing rough sets, which are a subject of this paper, were performed in order to confirm results obtained from the DM and ANN approaches. Another reason was to verify if all three computational intelligence models applied to the salience problem, return similar findings, which may prove the correctness of these approaches. This article is an extended version of a paper which is included in Proceedings of Rough Sets and Intelligent Systems Paradigms [12]. The remainder of the paper is organized as follows: in Section 2 a short review of computational intelligence methods that are used in research related to emulation of human perception is presented. Then, Section 3 shows some issues describing hypermetric rhythm retrieval, which direct towards the experiments on rhythm retrieval. A brief description of the application realizing automatic rhythm accompaniment is shown in Section 4 along with an approach to the computational complexity of the algorithm creating hypermetric rhythmic hypotheses (Section 5). Finally, Section 6 puts forward summary of results as well as some concluding remarks.

2 Emulation of Human Perception by Computational Intelligence Techniques

The domain of computational intelligence grows into independent and very attractive research area in a few last years, with many applications dedicated to data mining in musical domain [8], [9], [23], [24]. *Computational Intelligence* (CI) is a branch of *Artificial Intelligence*, which deals with the AI soft facets, i.e. programs behaving intelligently. The CI is understood in a number of ways, e.g. as a study of the design of intelligent agents or as a subbranch of AI, which aims "to use learning, adaptive, or evolutionary computation to create programs that are, in some sense, intelligent" [25]. Researchers are trying to classify the branches of CI to designate the ways in which CI methods help humans to discover how their perception works. However, this is a multi-facet task with numerous overlapping definitions, thus the map of this discipline is ambiguous. The domain of CI groups several approaches, the most common are: the Artificial Neural Networks (ANNs), Fuzzy Systems, Evolutionary Computation, Machine Learning including Data Mining, Soft Computing, Rough Sets, Bayesian Networks, Expert Systems and Intelligent Agents [18]. Currently, in the age of CI people are trying to build machines emulating human behaviors, and one of such applications concerns rhythm perception. This paper presents an example of how to design and build an algorithm which is able to emulate human perception of rhythm. Two CI approaches, namely the ANNs and Rough Sets (RS), are used in the experiments aiming at the estimation of musical salience. The first of them, the ANN model, concerns processes, which are not entirely known, e.g. human perception of rhythm. The latter is the RS approach, introduced by Pawlak [16] and used by many researches in data discovery and intelligent management [17], [18].

Since the applicability of ANNs in recognition was experimentally confirmed in a number of areas, neural networks are also used to estimate rhythmic salience of sounds. There exists a vast literature on ANNs, and for this reason only a brief introduction to this area is presented in this paper. A structure of an ANN usually employs the McCulloch-Pitts model, involving the modification of the neuron activation function, which is usually sigmoidal. All neurons are interconnected. Within the context of the neural network topology, ANNs can be classified as *feedforward* or *recurrent* networks, which are also called *feedback* networks. In the case of recurrent ANNs the connections between units form cycles, while in feedforward ANNs the information moves in only one direction, i.e. forward. The elements of a vector of object features constitute the values, which are fed to the input of an ANN. The type of data accepted at the input and/or returned at the output of an ANN is also a differentiating factor. The *quantitative* variable values are continuous by nature, and the *categorical* variables belong to a finite set (small, medium, big, large). The ANNs with continuous values at input are able to determine the degree of the membership to a certain class. The output of networks based on categorical variables may be Boolean, in which case the network decides whether an object belongs to a class or not. In the case of the salience problem the number of categorical output variables equals to two, and it is determined whether the sound is accented or not.

In the experiments the Authors examined whether a supervised categorical network such as Learning Vector Quantization (LVQ) is sufficient to resolve the salience problem. The classification task of the network was to recognize the sound as accented or not. LVQs are self-organizing networks with the ability to learn and detect the regularities and correlations at their input, and then to adapt their responses to that input. An LVQ network is trained in a supervised manner, it consists of the competitive and a linear layers. The first one classifies the input vectors into subclasses, and the latter transforms input vectors into target classes. On the other hand, the aim of the RS-based experiments was two-fold. First, it was to compare the results with the ones coming from the ANN. In addition, two schemes of data discretization were applied. In the case of k-means discretization accuracies of predictions are delivered.

3 Experiments

3.1 Database

Presented experiments were conducted on MIDI files of eighty national anthems, retrieved from the Internet. Storing information about meter in the files is necessary to indicate accented sounds in a musical piece. This information, however, is optional in MIDI files, thus information whether the sound is accented or not is not always available. In addition, in a number of musical files retrieved from the Internet, the assigned meter is incorrect or there is no information about meter at all. This is why the correctness of meter was checked by inserting an additional simple drum track into the melody. The hits of the snare drum were inserted in the locations of the piece calculated with Formula (1), where T is a period computed with the autocorrelation function, and i indicates subsequent hits of a snare drum.

$$i \cdot T, i = 0, 1, 2, \ldots \tag{1}$$

The Authors listened to the musical files with snare drum hits inserted, and rejected all the files in which accented locations were indicated incorrectly. Also some anthems with changes in time signature could not be included in the training and testing sets, because this metric rhythm retrieval method deals with hypotheses based on rhythmic levels of a constant period. Usually the change in time signature results in changes in the period of a rhythmic level corresponding to the meter, and an example of such change might be from 3/4 into 4/4. Conversely, an example of a change in time signature which does not influence the correct indication of accented sounds could be from 2/4 into 4/4. Salience experiments presented in this paper are conducted on polyphonic MIDI tracks containing melodies, overlapping sounds coming from the tracks other than melodic ones, were not included in the experimental sets.

For the purpose of the experiments the values of physical sounds' attributes were normalized and discretized with equal subrange technique. Minimum and maximum values within the domain of each attribute are found. The whole range is then divided into $m$subranges with thresholds between the subranges, placed in the locations counted with aid of the Formula (2).

$$MinValue + (MaxValue - MinValue) \cdot j/m \texttt{ for } j = 0, 1, 2, \ldots m \qquad (2)$$

3.2 ANN-Based Experiment

For the training phase, accented locations in each melody were found with methods described in Section 3.1. One of the tested networks had three separate inputs – one for each physical attribute of a sound (duration, frequency and amplitude - DPV). Three remaining networks had one input each. Each input took a different physical attribute of a given sound, namely D – duration, P – pitch (frequency) or V – velocity (amplitude). All attributes were from the range of 0 to 127. The network output was binary: 1 if the sound was accented, or 0 if it was not. Musical data were provided to the networks to train them to recognize accented sounds on the basis of physical attributes.

In this study LVQ network recognized a sound as 'accented' or 'not accented'. Since physical attributes are not the only features determining whether a sound is accented, some network answers may be incorrect. The network accuracy *NA* was formulated as the ratio of the number of accented sounds, which were correctly detected by the network, to the total number of accented sounds in a melody, as stated in Formula (3).

$$NA = \textit{number of accented sounds correctly detected by the network / number of all accented sounds} \qquad (3)$$

Hazard accuracy *HA* is the ratio of the number of accents given by the network to the number of all sounds in a set, as stated in Formula (4).

$$HA = \textit{number of accented sounds detected by the network / number of all sounds} \qquad (4)$$

The melodies of anthems were used to create 10 training/testing sets. Each set included 8 entire pieces. Each sound with an index divisible by 3 was assumed to be a training sound. The remaining sounds were treated as testing sounds. As a consequence, the testing set was twice as large as the training set. Accuracies in the datasets were averaged for each network separately. Evaluating a separate accuracy for each ANN allowed for comparing their preciseness. Standard deviations were also calculated. Fractions equal to standard deviations were divided by average values. Such fractions help compare the stability of results. The lower the value of the fraction is, the more stable the results are. All results are shown on the right side of Table 1. A single accuracy value was assigned to each ANN. Standard deviations were also calculated and the resultant stability fraction equal to standard deviations divided by average values was presented.

The accuracy of finding accented sounds estimated for four networks can be seen in Fig. 1, the plots are drawn on the basis of the data from Table 1. There

Table 1. Parameters of training and testing data and performance of ANNs

Set No.	Number of sounds			Acc./all	NA/HA			
	All	Accented	Not accented	[%]	*D*	*P*	*V*	*DPV*
1	937	387	550	41	1.90	1.01	0.95	1.96
2	1173	386	787	33	2.28	0.89	1.23	2.19
3	1054	385	669	37	2.14	0.96	0.11	2.13
4	937	315	622	34	2.25	1.13	0.79	2.49
5	801	293	508	37	1.98	1.02	1.04	1.95
6	603	245	358	41	1.67	1.02	0.93	1.24
7	781	332	449	43	1.93	0.98	1.16	1.89
8	880	344	536	39	2.06	0.97	1.13	2.14
9	867	335	532	39	1.91	0.87	0.83	1.73
10	1767	509	1258	29	2.14	0.72	1.62	2.66
Avg.	**980**	**353**	**626**	**37**	**2.03**	**0.96**	**0.98**	**2.03**
StdDev	317	71	251	4	0.19	0.11	0.39	0.39
StdDev/Avg					0.09	0.12	0.40	0.19

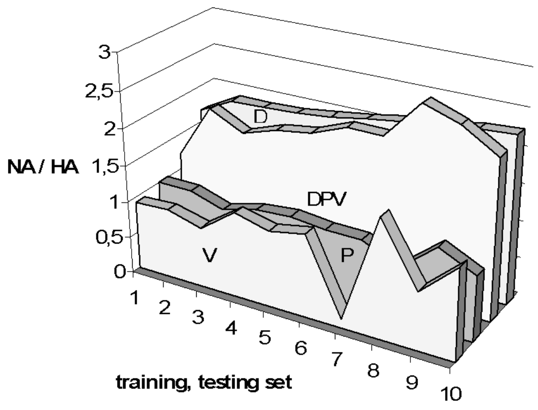


Fig. 1. Accuracy of four networks for melodies of anthems

are three plots presenting the results of networks fed with one attribute only, and one plot for the network presented with all three physical attributes at its single input (line *DPV*). The consequent pairs of training and testing sets are on the horizontal axis, the fraction *NA/HA*, signifying how many times an approach is more accurate than a blind choice, is on the vertical axis.

3.3 Rough Set-Based Experiments

The aim of this experiment was to obtain the results analogical to the ones coming from the ANN and to confront them with each other. In particular, it was expected to confirm whether physical attributes influence a tendency of sounds to be located in accented positions. Further, it was to answer how complex is the way the rhythmic salience of sound depends on its physical attributes, and to observe the stability of the accuracies obtained in the RS-based experiments.

In the rough set-based experiments, the dataset named RSESdata1 was split into training and testing sets in the 3:1 ratio. Then the rules were generated, utilizing a genetic algorithm available in the Rough Set Exploration System [1], [22]. For dataset RSESdata1, 7859 rules were obtained resulting in the classification accuracy of 0.75 with the coverage equal to 1. It should be remembered that accuracy is a measure of classification success, which is defined as a ratio of the number of properly classified new cases (objects) to the total number of new cases. Rules with support less than 10 were then removed. The set of rules was thus reduced to 427 and the accuracy dropped to 0.736 with the coverage still remaining 1. Then the next attempt to further decrease the number of rules was made, and rules with support less than 30 were excluded. In this case, 156 rules were still valid but the accuracy dropped significantly, i.e. to 0.707, and at the same time the coverage fall to 0.99. It was decided that for a practical implementation of a rough set-based classifier, a set of 427 rules is suitable. Reducts used in rule generation are presented in Table 2.

The same approach was used for dataset RSESdata2, and resulted in 11121 rules with the accuracy of 0.742 and the coverage of 1. After removing rules with support less than 10, only 384 rules remained, and the accuracy dropped to 0.735. Again, such number of rules is practically applicable. Reducts used in rule generation are presented in Table 3.

The approach taken to LVQ network was also implemented for rough sets. Ten different training\test sets were acquired by randomly splitting data into

Table 2. Reduct for RSESdata1 dataset

Reducts	Positive Region	Stability Coefficient
{ duration, pitch }	0.460	1
{ duration, velocity }	0.565	1
{ pitch, velocity }	0.369	1
{ duration }	0.039	1
{ pitch }	0.002	1
{ velocity }	0.001	1

Table 3. Reduct for RSESdata2 dataset

Reducts	Positive Region	Stability Coefficient
{ duration, velocity }	0.6956	1
{ duration, pitch }	0.6671	1
{ pitch, velocity }	0.4758	1
{ duration }	0.0878	1
{ pitch }	0.0034	1
{ velocity }	0.0028	1

Table 4. Parameters of training and testing data and performance of RSES (RSA is a Rough Set factor, analogical to *NA* in ANNs)

Set No.	Number of sounds			Acc/all	RSA/HA			
	All testing sounds	Accented	Not accented		*D*	*P*	*V*	*DPV*
1	1679	610	1069	36.33	1.81	1.06	1.21	1.75
2	1679	608	1071	36.21	1.90	1.08	1.09	1.74
3	1679	594	1085	35.37	1.84	1.12	1.19	1.74
4	1679	638	1041	37.99	1.68	1.08	1.12	1.62
5	1679	632	1047	37.64	1.67	1.07	1.12	1.64
6	1679	605	1074	36.03	1.87	1.16	1.13	1.88
7	1679	573	1106	34.12	1.77	1.09	1.18	1.68
8	1679	618	1061	36.80	1.90	1.06	1.17	1.73
9	1679	603	1076	35.91	1.77	1.08	1.11	1.70
10	1679	627	1052	37.34	1.77	1.08	1.15	1.66
Avg.	**1679**	**610**	**1068**	**36.37**	**1.80**	**1.09**	**1.15**	**1.72**
StdDev	0	19.2	19.2	1.14	0.08	0.02	0.039	0.07
StdDev/Avg					0.04	0.02	0.033	0.04

five pairs, and than each set in a pair was further divided into two sets – a training and a testing one – with the 2:1 ratio. Therefore testing sets contained 1679 objects each. The experiments, however, were based on RSESdata1 set because of its higher generalization ability (see Table 4).

It should be remembered that reduct is a set of attributes that discerns objects with different decisions. Positive region shows what part of indiscernibility classes for a reduct is inside the rough set. The larger boundary regions are, the more rules are nondeterministic, and the smaller positive region is. Stability coefficient reveals if the reduct appears also for subsets of original dataset, which are calculated during the reduct search. For reduct {duration} positive region is very small, but during classification a voting method is used to infer correct outcome from many nondeterministic rules, and, finally, high accuracy is obtained. Adding another dimension, e.g. {duration, velocity}, results in higher number of deterministic rules, larger positive region, but it does not guarantee the accuracy increase (Table 4).

Rules were generated utilizing different reduct sets (compare with Table 1):

D - {duration} only; *P* - {pitch} only; *V* - {velocity} only; *DPV* - all 6 reducts {duration, velocity}, {duration, pitch}, {pitch, velocity}, {duration}, {pitch}, {velocity} have been employed.

***k*-NN Discretization.** The data were analyzed also employing *k*-NN method, which is implemented as a part of the RSES system [22]. The experiment was carried out differently in comparison to previously performed experiments using ANN (LVQ) and RS. The reason for this was to observe accuracy of classification while various number of *k* values has been set. It may easily be observed that lower number of clusters implies better accuracy of the predictions and a smaller

Table 5. Cut points in the case of k=3

Duration	Pitch	Velocity
45.33	44.175	44.909
133.88	78.285	75.756

Table 6. Classification results for k=3

	1	0	No. of obj.	Accuracy	Coverage
1	665	206	899	0.763	0.969
0	375	1,190	1,570	0.76	0.997
True positive rate	0.64	0.85			

Table 7. Cut points in the case of k=4

Duration	Pitch	Velocity
38.577	25.622	41.18
98.989	51.85	65.139
198.56	79.988	89.67

Table 8. Classification results for k=4

	1	0	No. of obj.	Accuracy	Coverage
1	640	220	899	0.744	0.957
0	353	1,202	1,570	0.773	0.99
True positive rate	0.64	0.85			

number of rules generated. In the following experiments full attributes vectors [“*duration*”, “*pitch*”, ”*velocity*”] are used as reducts. The k-means discretization is performed, where k values are set manually k={4, 5,10,15,20}. For a given k exactly k clusters are calculated, represented by their center points. The cut point is set as a middle point between two neighbor cluster centers. Cut points are used for attribute discretization and then rough set rules are generated. The training set comprises 7407 objects and the testing one 2469 objects (3:1 ratio).

Experiment I – k-means discretization (k=3) of each attribute (“*duration*”, “*pitch*” ,”*velocity*”), 872 rules. Cut points are shown in Table 5 and classification results in Table 6 (Total accuracy: 0.761, total coverage: 0.987).

Experiment II – k-means discretization (k=4) of each attribute (“*duration*”, “*pitch*” ,”*velocity*”), 1282 rules. Cut points are shown in Table 7 and classification results in Table 8 (Total accuracy: 0.763; total coverage: 0.978).

Experiment III – k-means discretization (k=5) of each attribute (“*duration*”, “*pitch*” ,”*velocity*”), 1690 rules. Cut points are shown in Table 9 and classification results in Table 10 (Total accuracy: 0.766: total coverage: 0.967).

Table 9. Cut points in the case of k=5

Duration		Pitch		Velocity	
31.733	133.91	24.224	68.629	27.826	66.759
72.814	259.15	47.536	94.84	48.708	89.853

Table 10. Classification results for k=5

	1	0	No. of obj.	Accuracy	Coverage
1	619	232	899	0.727	0.947
0	326	1,211	1,570	0.788	0.979
True positive rate	0.66	0.84			

Table 11. Cut points in the case of k=10

Duration		Pitch		Velocity	
11.11	121.62	18.089	73.161	14.558	57.81
27.319	174.66	35.259	82.648	27.307	67.94
44.375	264.32	47.046	95.963	36.02	81.992
62.962	642.94	56.279	119.15	42.846	102.45
86.621		64.667		49.768	

Table 12. Classification results for k=10

	1	0	No. of obj.	Accuracy	Coverage
1	533	227	899	0.701	0.845
0	253	1,162	1,570	0.821	0.901
True positive rate	0.68	0.84			

Table 13. Cut points in the case of k=15

Duration		Pitch		Velocity	
3.2372	58.942	9.4427	64.17	15.139	56.751
8.8071	76.842	23.023	70.125	28.128	60.378
13.854	101.33	33.148	74.465	37.217	63.963
20.693	132.65	41.285	79.687	44.091	68.747
29.284	183.95	48.225	86.909	49.015	75.963
37.857	283.39	53.264	97.988	52.011	86.914
46.806	656.29	57.875	119.79	53.913	104.36

Experiment IV – k-means discretization (k=10) of each attribute ("*duration*", "*pitch*" ," *velocity*"), 2987 rules. Cut points are shown in Table 11 and classification results in Table 12 (Total accuracy: 0.779: total coverage: 0.881).

Table 14. Classification results for k=15

	1	0	No. of obj.	Accuracy	Coverage
1	492	217	899	0.694	0.789
0	229	1,121	1,570	0.83	0.86
True positive rate	0.68	0.84			

Table 15. Cut points in the case of k=20

Duration			Pitch			Velocity		
2.920	38.505	107.31	7.584	41.5	65.348	9.603	61.725	78.691
7.388	45.68	132.67	18.309	42.674	70.631	22.588	65.283	83.434
11.601	53.632	173.43	25.782	44.977	76.774	33.807	68.596	89.438
16.635	63.169	235.67	31.629	49.35	84.981	43.158	71.363	96.862
21.486	74.84	326.34	35.553	54.714	97.402	50.163	73.564	106.57
26.464	88.631	672.55	38.184	60.045	119.79	55.096	75.624	121.49
31.96			40.174			58.528		

Table 16. Classification results for k=20

	1	0	No. of obj.	Accuracy	Coverage
1	476	223	899	0.681	0.778
0	233	1,122	1,570	0.828	0.863
True positive rate	0.67	0.83			

Experiment V – k-means discretization (k=15) of each attribute ("*duration*", "*pitch*" ," *velocity*"), 3834 rules. Cut points are shown in Table 13 and classification results in Table 14 (Total accuracy: 0.783: total coverage: 0.834).

Experiment VI – k-means discretization (k=20) of each attribute ("*duration*", "*pitch*" ," *velocity*"), 4122 rules. Cut points are shown in Table 15 and classification results in Table 16 (Total accuracy: 0.778: total coverage: 0.832).

Retrieving rhythmical patterns together with their hierarchical structure of rhythm acquired with machine learning is a step towards an application capable of creating automatic drum accompaniment to a given melody. Such a computer system is to be presented in Section 4.

4 Automatic Drum Accompaniment Application

The hypermetric rhythm retrieval approach proposed in this article is illustrated with a practical application of a system automatically generating a drum accompaniment to a given melody. A stream of sounds in MIDI format is introduced at the system input, on the basis of a musical content the method retrieves a hypermetric structure of rhythm of a musical piece consisting rhythmic motives, phrases, and sentences. A method does not use any information about rhythm

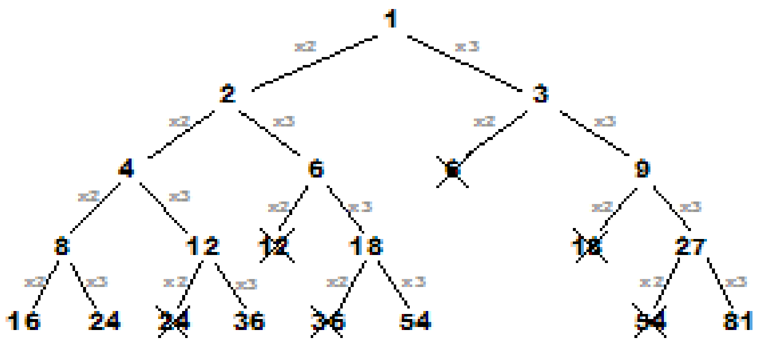


Fig. 2. The tree of periods

(time signature), which is present often in MIDI files. Neither rhythmic tracks nor harmonic information are used to support the method. The only information analyzed is a melody, which might be monophonic as well as polyphonic. Two elements are combined, namely recurrence of melodic and rhythmic patterns and the rhythmic salience of sounds to create a machine able to find the metric structure of rhythm to a given melody.

The method proposed by the authors of this paper generates rhythmic levels first, and then uses these levels to compose rhythmic hypotheses. The lowest rhythmic level has a phase of the first sound from the piece and its period is atomic. The following levels have periods of values achieved by recursive multiplication of periods that have already been calculated (starting from the atomic value) by the most common prime numbers in Western music, i.e. 2 and 3. The process of period generation may be illustrated as a process of a tree structure formation (Figure 2) with a root representing the atomic period equal to 1. Each node is represented by a number which is the node ancestor number multiplied by either 2 or 3.

The tree holds some duplicates. The node holding a duplicated value would generate a sub-tree whose all nodes would also be duplicates of already existing values. Thus duplicate subtrees are eliminated and we obtain a graphical interpretation in the form of the period triangle (see Figure 3), where the top row refers to a quarter-note, and consecutively to a half-note, whole note (motive), phrase, sentence and period.

When the phase of periods creation is completed, each period must have all its phases (starting from phase 0) generated. The last phase of a given rhythmic level has the value equal to the size of the period decreased by one atomic period. In order to achieve hypotheses from the generated rhythmic levels, it is necessary to find all families of related rhythmic levels. A level may belong to many levels. The generated hypotheses are instantly ranked to extract the one which designates the appropriate rhythm of the piece. The hypotheses that cover notes of significant rhythmic weights are ranked higher. The weights are calculated based on the knowledge gathered by learning systems that know how to asses the importance of physical characteristics of sounds that comprise the

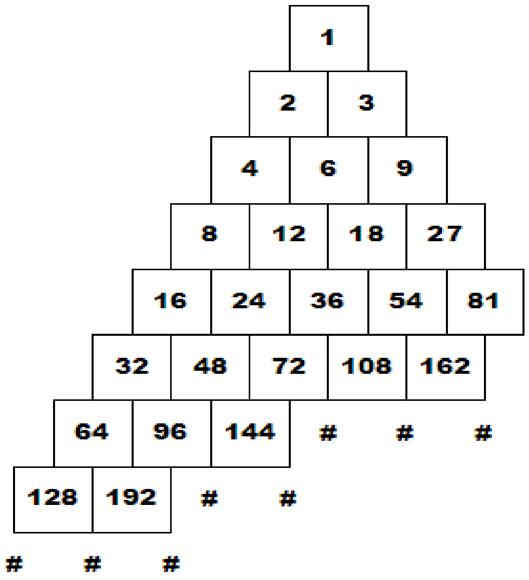


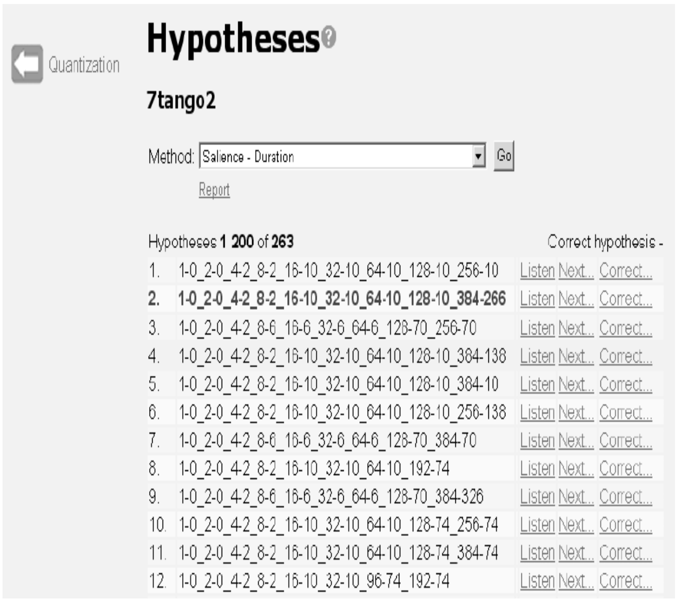
Fig. 3. Triangle of periods

Table 17. Drum instruments added at a particular rhythmic level

Rhythmic level	Name of the instrument
1	Closed hi-hat
2	Bass drum
3	Snare drum
4	Open triangle
5	Splash cymbal
6	Long whistle
7	Chinese cymbal

piece. The system proposed by the authors employs rules obtained in the process of *data mining* [11], [26], as well as from the operation of neural networks [6], and through employing rough sets [12]. Taking a set of representative musical objects as grounds, these systems learn how to asses the influence of a sound relative frequency, amplitude and length on its rhythmic weight. The second group of methods used to rank hypotheses is based on one of the elementary rules known in music composition, i.e. recurrence of melodic and rhythmic patterns – the group is described in the authors' works [10], [27].

The application realizing an automatic accompaniment, called DrumAdd, accepts a MIDI file at its input. The accompaniment is added to the melody by inserting a drum channel, whose number is 10 in the MIDI file. Hi-hat hits are inserted in the locations of rhythmic events associated with the first rhythmic



Hypotheses

7tango2

Method: Salience - Duration Go

Report

Hypotheses **1 200** of **263** — Correct hypothesis -

1.	1-0_2-0_4-2_8-2_16-10_32-10_64-10_128-10_256-10	Listen	Next...	Correct...
2.	**1-0_2-0_4-2_8-2_16-10_32-10_64-10_128-10_384-266**	Listen	Next...	Correct...
3.	1-0_2-0_4-2_8-6_16-6_32-6_64-6_128-70_256-70	Listen	Next...	Correct...
4.	1-0_2-0_4-2_8-2_16-10_32-10_64-10_128-10_384-138	Listen	Next...	Correct...
5.	1-0_2-0_4-2_8-2_16-10_32-10_64-10_128-10_384-10	Listen	Next...	Correct...
6.	1-0_2-0_4-2_8-2_16-10_32-10_64-10_128-10_256-138	Listen	Next...	Correct...
7.	1-0_2-0_4-2_8-6_16-6_32-6_64-6_128-70_384-70	Listen	Next...	Correct...
8.	1-0_2-0_4-2_8-2_16-10_32-10_64-10_192-74	Listen	Next...	Correct...
9.	1-0_2-0_4-2_8-6_16-6_32-6_64-6_128-70_384-326	Listen	Next...	Correct...
10.	1-0_2-0_4-2_8-2_16-10_32-10_64-10_128-74_256-74	Listen	Next...	Correct...
11.	1-0_2-0_4-2_8-2_16-10_32-10_64-10_128-74_384-74	Listen	Next...	Correct...
12.	1-0_2-0_4-2_8-2_16-10_32-10_96-74_192-74	Listen	Next...	Correct...

Fig. 4. User interface of an automatic drum accompaniment application

level. The consecutive drum instruments associated with higher rhythmic levels are: bass drum, snare drum, open triangle, splash cymbal, long whistle and a Chinese cymbal, as it is shown in Table 17.

The DrumAdd system was developed in Java. The main window of the system can be seen in Figure 4 – the user interface shown is entitled 'Hypotheses'. Default settings of quantization are as follows:

- onsets of sounds are shifted to time grid of one-eight note,
- durations of sounds are natural multiplies of one-eight note,
- notes shorter than one-sixteenth note are deleted.

A user may easily change quantization settings. A hypothesis ranking method can be chosen in a drop-down list ('Salience - Duration' in the case presented). A user may listen to the accompaniment made on the basis of the hypothesis (link 'Listen'), change the drum sounds associated to the consecutive rhythmic levels (link 'Next...') or acknowledge the given hypothesis as correct (link 'Correct' is assumed to be correct). A user also receives an access to report and ranking of hypotheses, which presents a table with accuracies corresponding to hypotheses ranking methods.

The drum Accompaniment is generated automatically to the sample melodies contained in the system. As a result some sample pieces contain a drum track created strictly with the approach presented earlier. In the second group of examples, the accompaniment is created on the basis of a metric structure retrieved automatically.

5 Algorithm Complexity

This section addresses the problem of computational algorithm complexity. Three phases of the algorithm engineered by the authors, namely creating periods, simplified hypotheses and full hypotheses are examined. The analyses of computational complexity of the method proposed assume that the engineered method is expected to rank rhythmic hypotheses formed of three rhythmic levels above meter. This proved to be sufficient for providing automatic drum accompaniment for a given melody without delay. The method creates all possible rhythmic structures. However, their number is limited and depends on the following factors [28]:

- The level designated as the lowest among all the created hypotheses (this defines the parameter of sound length quantization). The authors observed that the quantization with the resolution of a quarter-note is sufficient.
- The intricacy of the hypotheses, i.e. how many levels they contain. The method was examined for at most three rhythmic levels above meter, similarly as in the research conducted by Rosenthal [20], and Temperley and Sleator [21].

Taking the above assumptions into consideration, i.e. the quantization parameter being a quarter-note and the analysis of a hypothesis concerning three levels above meter, we obtain the number of periods from the first 6 layers of the triangle shown in Figure 3. The atomic period is a quarter-note (layer 1), the layer containing periods 4, 6, 9 is the level of meter, and the sixth layer holding the values of 32, 48, 72 ... is the last examined rhythmic level.

Calculating periods. The number of periods is $n{\cdot}(n{+}1)/2$, where n is the number of layers, so the algorithm is polynomial, the function of the computational complexity is of class $\mathrm{O(n^2)}$. The basic operation that calculates periods is multiplication. The number of periods calculated for 6 layers is 21, and these are the elements of a period list.

Creating hypotheses. Hypotheses (with periods only) are lists of related rhythmic levels that include pairs of <period, phase> values. If we take only periods into consideration, the number of hypotheses is the number of paths starting from the highest rhythmic level (layer 6) and ending in the level of atomic period (layer 1). For assumed parameters, this gives 32 hypotheses if only periods defined. The number is a result of the following computations:

- from period 32 there is one path (32, 16, 8, 4, 2, 1,),
- from period 48 there are 5 paths,
- from period 72 there are 10 paths,

For the left half of the triangle we may specify 16 paths. The computations for the right half, i.e. the paths including periods 108, 162, and 243, are analogous. This gives 32 paths altogether in a 6 layer triangle.

The function of computational complexity is of class $\mathrm{O(n^2)}$, where n is the number of layers. Thus, the complexity is exponential which with n limited to

Table 18. Rhythmic hypotheses (without phases) for a 6-layer triangle of periods

Layer	**1**	**2**	**3**	**4**	**5**	**6**
	1	2	4	8	16	32
	1	2	4	8	16	48
	1	2	4	8	24	48
	1	2	4	8	24	72
	1	2	4	12	24	48
	1	2	4	12	24	72
	1	2	4	12	36	72
	1	2	4	12	36	108
	1	2	6	12	24	48
	1	2	6	12	24	72
	1	2	6	12	36	72
	1	2	6	12	36	108
	1	2	6	18	36	72
	1	2	6	18	36	108
	1	2	6	18	54	108
	1	2	6	18	54	162
	1	3	6	12	24	48
	1	3	6	12	24	72
	1	3	6	12	36	72
	1	3	6	12	36	108
	1	3	6	18	36	72
	1	3	6	18	36	108
	1	3	6	18	54	108
	1	3	6	18	54	162
	1	3	9	18	36	72
	1	3	9	18	36	108
	1	3	9	18	54	108
	1	3	9	18	54	162
	1	3	9	27	54	108
	1	3	9	27	54	162
	1	3	9	27	81	162
	1	3	9	27	81	243

6 layers confines the number of hypotheses to 32. The rows of Table 6 show subsequent simplified hypotheses, i.e. the ones that contain only periods (phases are ignored) for the example from Figures 2 and 3.

The algorithm that creates hypotheses with periods only ranks rhythmic hypotheses based on the recurrence of melorhythmic patterns (16 methods proposed in the thesis of Wojcik [27]). The basic operation of patterns recurrence evaluation is in this case addition. The only hypotheses ranking method examined by the authors that requires the phases to be defined is the method based on rhythmic weights.

Creating hypotheses with phases. Each hypotheses may have as many versions, with regard to phases, as its longest period is, e.g. the first hypothesis from Table 17 (the first row: 1, 2, 4, 8, 16, 32) will have 32 different phases. On condition that $n = 6$, the number of all hypotheses for the discussed example will amount to 3125, which is the sum of all periods from layer 6. Thus, the number of all hypotheses is the sum of the values from the last column of Table 18.

The algorithm that forms hypotheses with phases is used in a method ranking rhythmic hypotheses based on rhythmic weight. The elementary operation of this method is addition.

To analyze a piece of music with regard to its motives, phases, phrases and periods when its atomic period is defined as a quarter-note, the number of 6 layers (n=6) is sufficient. Despite the exponential complexity of the method, the number of elementary operations is not more than 10^4on a 1.6 GHz computer. The total time of all operations for a single piece of music is imperceptible for a system user, which was proved by the experimental system, engineered by the authors. This means that the method provides high quality automatic drum accompaniment without a delay.

6 Concluding Remarks

Employing computational approach is helpful in retrieving the time signature and the locations of barlines from a piece on the basis of its content only. Rhythmic salience approach worked out and described in this paper may also be valuable in ranking rhythmic hypotheses and music transcription. A system, creating drum accompaniment to a given melody, automatically, on the basis of highly ranked rhythmic hypothesis is a useful practical application of rhythmic salience method. A prototype of such a system, using salience approach was developed on the basis of findings of authors of this paper, and it works without delay, even though its computational complexity is quite considerable.

On the basis of the results (see Tables 1, 4) obtained for both: RS and ANN experiments, it may be observed that the average accuracy of all approaches taking duration D into account – solely or in the combination of all three attributes DPV – is about twice as good as hazard accuracy (values of 1.72 for Rough Set DPV, 1.80 for Rough Set D, and a value of 2.03 both for Network D and for Network DPV were achieved). The performance of approaches considering pitch P and velocity V separately are very close to random accuracy, the values are equal to 1.09 and 1.15 for Rough Sets. For the ANN, the values are 0.96 and 0.98, respectively. Thus, it can be concluded that the location of a sound depends only on its duration.

The algorithms with the combination of DPV attributes performed as well as the one based only on duration, however this is especially valid for ANNs, rough sets did a little bit worse. Additional attributes do not increase the performance of the ANN approach. It can be thus concluded that the rhythmic salience depends on physical attributes in a simple way, namely it depends on a single physical attribute – duration.

Network *D* is the ANN that returns the most stable results. The value of fraction in the third row of Table 1 is low for this network and it is equal to 0.09. Network *DPV*, which takes all attributes into account, is much less reliable because the stability fraction is about twice worse than the stability of Network *D* and it is equal to 0.19. The stability of Network *P*, considering the pitch, is quite high (it equals 0.12), but its performance is close to the random choice. For learning and testing data used in this experiment, velocity appeared to be the most data-sensitive attribute (see results of Network *V*). Additionally, this network appeared to be unable to find accented sounds.

In the case of Rough Sets, the duration-based approaches *D* and *DPV* returned less stable results than *P* and *V* approaches. Values of 0.045, 0.043, 0.026, 033 were obtained for *D*, *DPV*, *P*, and *V* respectively.

The ANN salience-based experiments described in the earlier work by the Authors [7], were conducted on a database of musical files containing various musical genres. It consisted of monophonic (non-polyphonic), and the polyphonic files. Also, a verification of the association rules model of the Data Mining domain for musical salience estimation was presented in that paper. The conclusions derived from the experiments conducted on national anthems for the purpose of this paper, are consistent with the ones described in the work by Kostek et al. [7]. Thus, the ANNs can be used in systems of musical rhythm retrieval in a wide range of genres and regardless of the fact whether the music is monophonic of polyphonic. The average relative accuracy for duration-based approaches where Rough Sets are used is lower than this obtained by LVQ ANNs. However, the same tendency is noticeable – utilization of the duration parameter leads to successful classification. The *P* (pitch) and *V* (velocity) parameters appeared not to be important in making decision about rhythmical structure of a melody. Finally, using different discretization schemes instead of the equal subrange technique does not change the accuracy of rough sets-based rhythm classification, significantly.

Acknowledgments

The research was partially supported by the Polish Ministry of Science and Education within the project No. PBZ-MNiSzW-02/II/2007.

References

1. Bazan, J.G., Szczuka, M.S.: The Rough Set Exploration System. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets III. LNCS, vol. 3400, pp. 37–56. Springer, Heidelberg (2005)
2. van Belle, W.: BPM Measurement of Digital Audio by means of Beat Graphs & Ray Shooting. Department Computer Science, University Tromsø (Retrieved, 2004), `http://bio6.itek.norut.no/werner/Papers/bpm04/`
3. Dahl, S.: On the beat - Human movement and timing in the production and perception of music. Ph.D. Thesis, KTH Royal Institute of Technology, Stockholm, Sweden (2005)

4. Dixon, S.: Automatic Extraction of Tempo and Beat from Expressive Performances. J. of New Music Research 30(1), Swets & Zeitlinger, 39–58 (2001)
5. Huron, D.: Review of Harmony: A psychoacoustical Approach (Parncutt, 1989); Psychology of Music 19(2), 219–222 (1991)
6. Kostek, B., Wójcik, J.: Machine Learning System for Estimation Rhythmic Salience of Sounds. Int. J. of Knowledge-Based and Intelligent Engineering Systems 9, 1–10 (2005)
7. Kostek, B., Wójcik, J., Holonowicz, P.: Estimation the Rhythmic Salience of Sound with Association Rules and Neural Networks. In: Proc. of the Intern. IIS: IIPWM 2005, Intel. Information Proc. and Web Mining, Advances in Soft Computing, pp. 531–540. Springer, Sobieszewo (2005)
8. Kostek, B.: Perception-Based Data Processing in Acoustics. In: Applications to Music Information Retrieval and Psychophysiology of Hearing. Series on Cognitive Technologies. Springer, Heidelberg (2005)
9. Kostek, B.: Applying computational intelligence to musical acoustics. Archives of Acoustics 32(3), 617–629 (2007)
10. Kostek, B., Wójcik, J.: Automatic Retrieval of Musical Rhythmic Patterns, vol. 119. Audio Engineering Soc. Convention, New York (2005)
11. Kostek, B., Wójcik, J.: Automatic Salience-Based Hypermetric Rhythm Retrieval. In: International Workshop on Interactive Multimedia and Intelligent Services in Mobile and Ubiquitous Computing, Seoul, Korea. IEEE CS, Los Alamitos (2007)
12. Kostek, B., Wójcik, J., Szczuko, P.: Searching for Metric Structure of Musical Files. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 774–783. Springer, Heidelberg (2007)
13. Lerdahl, F., Jackendoff, R.: A Generative Theory of Tonal Music. MIT Press, Cambridge (1983)
14. McAuley, J.D., Semple, P.: The effect of tempo and musical experience on perceived beat. Australian Journal of Psychology 51(3), 176–187 (1999)
15. Parncutt, R.: Harmony: A Psychoacoustical Approach. Springer, Berlin (1989)
16. Pawlak, Z.: Rough Sets. Internat. J. Computer and Information Sciences 11, 341–356 (1982)
17. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Information Sciences 177, 3–27 (2007)
18. Peters, J.F., Skowron, A. (eds.): Transactions on Rough Sets V. LNCS, vol. 4100. Springer, Heidelberg (2004-2008)
19. Rosenthal, D.F.: Emulation of human rhythm perception. Comp. Music J. 16(1), 64–76 (Spring, 1992)
20. Rosenthal, D.F.: Machine Rhythm: Computer Emulation of Human Rhythm Perception, Ph.D. thesis. MIT Media Lab, Cambridge, Mass. (1992)
21. Temperley, D., Sleator, D.: Modeling meter and harmony: A preference-rule approach. Comp. Music J. 15(1), 10–27 (1999)
22. RSES Homepage, `http://logic.mimuw.edu.pl/~rses`
23. Wieczorkowska, A., Czyzewski, A.: Rough Set Based Automatic Classification of Musical Instrument Sounds. Electr. Notes Theor. Comput. Sci. 82(4) (2003)
24. Wieczorkowska, A., Raś, Z.W.: Editorial: Music Information Retrieval. J. Intell. Inf. Syst. 21(1), 5–8 (2003)
25. Wikipedia homepage
26. Wójcik, J., Kostek, B.: Intelligent Methods for Musical Rhythm Finding Systems. In: Nguyen, N.T. (ed.) Intelligent Technologies for Inconsistent Processing. International Series on Advanced Intelligence, vol. 10, pp. 187–202 (2004)

27. Wójcik, J.: Methods of Forming and Ranking Rhythmic Hypotheses in Musical Pieces, Ph.D. Thesis, Electronics, Telecommunications and Informatics Faculty, Gdansk Univ. of Technology, Gdansk (2007)
28. Wójcik, J., Kostek, B.: Computational Complexity of the Algorithm Creating Hypermetric Rhythmic Hypotheses. Archives of Acoustics 33(1), 57–63 (2008)

FUN: Fast Discovery of Minimal Sets of Attributes Functionally Determining a Decision Attribute

Marzena Kryszkiewicz and Piotr Lasek

Institute of Computer Science, Warsaw University of Technology
Nowowiejska 15/19, 00-665 Warsaw, Poland
{mkr,p.lasek}@ii.pw.edu.pl

Abstract. In this paper, we present our *Fun* algorithm for discovering minimal sets of conditional attributes functionally determining a given dependent attribute. In particular, the algorithm is capable of discovering Rough Sets certain, generalized decision, and membership distribution reducts. *Fun* can operate either on partitions of objects or alternatively on stripped partitions, which do not store singleton groups. It is capable of using functional dependencies occurring among conditional attributes for pruning candidate dependencies. In this paper, we offer further reduction of stripped partitions, which allows correct determination of minimal functional dependencies provided optional candidate pruning is not carried out. In the paper we consider six variants of *Fun*, including two new variants using reduced stripped partitions. We have carried out a number of experiments on benchmark data sets to test the efficiency of all variants of *Fun*. We have also tested the efficiency of the *Fun*'s variants against the Rosetta and RSES toolkits' algorithms computing all reducts and against *Tane*, which is one of the most efficient algorithms computing all minimal functional dependencies. The experiments prove that *Fun* is up to 3 orders of magnitude faster than the the Rosetta and RSES toolkits' algorithms and faster than *Tane* up to 30 times.

Keywords: Rough Sets, information system, decision table, reduct, functional dependency.

1 Introduction

The determination of minimal functional dependencies is a standard task in the area of relational databases. Tane [6] or Dep-Miner [14] are example efficient algorithms for discovering minimal functional dependencies from relational databases. A variant of the task, which consists in discovering minimal sets of conditional attributes that functionally or approximately determine a given decision attribute, is one of the topics of Artificial Intelligence and Data Mining. Such sets of conditional attributes can be used, for instance, for building classifiers. In the terms of Rough Sets, such minimal conditional attributes are called reducts [18]. One can distinguish a number of types of reducts. Generalized decision reducts (or equivalently, possible/approximate reducts [9]), membership distribution reducts (or equivalently, membership reducts [9]), and certain decision reducts belong to most popular Rough Sets reducts. In general, these types of reducts do not determine the decision attribute functionally. However, it was shown in [10] that these types of reducts are minimal sets of conditional attributes functionally determining appropriate modifications of the decision attribute. Thus, the task of searching such reducts is equivalent to looking for minimal sets of attributes functionally determining a given attribute. In this paper, we focus on finding all such minimal sets of attributes. To this end, one might consider applying either methods for discovering Rough Sets reducts, or discovering all minimal functional dependencies and then selecting such that determine a requested attribute.

A number of methods for discovering different types of reducts have already been proposed in the literature. e.g. [3-5],[7-8],[11-12],[15-29]. The most popular methods are based on discernibility matrices [21]. Unfortunately, the existing methods for discovering all reducts are not scalable. The recently offered algorithms for finding all minimal functional dependencies are definitely faster. In this paper, we focus on direct discovery of all minimal functional dependencies with a given dependent attribute, and expect this process to be faster than the discovery of all minimal functional dependencies. First, we present efficient *Fun* algorithm, we offered recently [12]. *Fun* discovers minimal functional dependencies with a given dependent attribute, and, in particular, is capable of discovering three above mentioned types of reducts. *Fun* can operate either on partitions of objects or alternatively on stripped object partitions, which do not store singleton groups. It is capable of using functional dependencies occurring among conditional attributes, which are found as a side effect, for pruning candidate dependencies.

In this paper, we extend our proposal from [12]. We offer further full and partial reduction of stripped partitions, which allows correct determination of minimal functional dependencies provided optional candidate pruning is not carried out. Then, we compare the efficiency of two new variants of *Fun* and four other variants of this algorithm, we proposed in [12]. We also test the efficiency of the *Fun*'s variants against the Rosetta and RSES toolkits' algorithms computing all reducts and against *Tane*, which is one of the most efficient algorithms computing all minimal functional dependencies.

The layout of the paper is as follows: Basic notions of information systems, functional dependencies, decision tables and reducts are recalled in Section 2. In Section 3, we present the *Fun* algorithm. Entirely new contribution is presented in subsection 3.5, where we describe how to reduce stripped partitions and provide two new variants of the *Fun* algorithm. The experimental evaluation of 6 variants of *Fun*, as well as the Rosetta and RSES toolkits' algorithms and *Tane*, are reported in Section 4. We conclude our results in Section 5.

2 Basic Notions

2.1 Information Systems

An *information system* is a pair $S = (O, AT)$, where O is a non-empty finite set of *objects* and AT is a non-empty finite set of *attributes* of these objects. In the sequel, $a(x)$, $a \in AT$ and $x \in O$, denotes the value of attribute a for object x, and V_a denotes the *domain* of a. Each subset of attributes $A \subseteq AT$ determines a binary A-*indiscernibility* relation $IND(A)$ consisting of pairs of objects indiscernible wrt. attributes A; that is, $IND(A) = \{(x, y) \in O \times O | \forall_{a \in A}\, a(x) = a(y)\}$. $IND(A)$ is an equivalence relation and determines a partition of O, which is denoted by π_A. The set of objects indiscernible with an object x with respect to A in S is denoted by $I_A(x)$ and is called A-*indiscernibility class*; that is, $I_A(x) = \{y \in O | (x, y) \in IND(A)\}$. Clearly, $\pi_A = \{I_A(x) | x \in O\}$.

Table 1. Sample information system $S = (O, AT)$, where $AT = \{a, b, c, e, f\}$

oid	*a*	*b*	*c*	*e*	*f*
1	1	0	0	1	1
2	1	1	1	1	2
3	0	1	1	0	3
4	0	1	1	0	3
5	0	1	1	2	2
6	1	1	0	2	2
7	1	1	0	2	2
8	1	1	0	2	2
9	1	1	0	3	2
10	1	0	0	3	2

Example 2.1.1. Table 1 presents a sample information system $S = (O, AT)$, where O is the set of ten objects and $AT = \{a, b, c, e, f\}$ is the set of attributes of these objects. $\square$

2.2 Functional Dependencies

Functional dependencies are of high importance in designing relational databases. We recall this notion after [2]. Let $S = (O, AT)$ and $A, B \subseteq AT$. $A \to B$ is defined a *functional dependency* (or A is defined to *determine* B *functionally*), if $\forall_{x \in O}\, I_A(x) \subseteq I_B(x)$. A functional dependency $A \to B$ is called *minimal*, if $\forall_{C \subset A}\, C \to B$ is not functional.

Example 2.2.1. Let us consider the information system in Table 1. $\{ce\} \to \{a\}$ is a functional dependency, nevertheless, $\{c\} \to \{a\}$, $\{e\} \to \{a\}$, and $\emptyset \to \{a\}$ are not. Hence, $\{ce\} \to \{a\}$ is a minimal functional dependency. $\square$

Property 2.2.1. Let $A, B, C \subseteq AT$.

a) If $A \to B$ is a functional dependency, then $\forall_{C \supset A}\, C \to B$ is functional.
b) If $A \to B$ is not functional, then $\forall_{C \subset A}\, C \to B$ is not functional.
c) If $A \to B$ is a functional dependency, then $\forall_{C \supset A}\, C \to B$ is a non-minimal functional dependency.
d) If $A \to B$ and $B \to C$ are functional dependencies, then $A \to C$ is a non-minimal functional dependency.
e) If $A \subset B$ and $B \cap C = \emptyset$, then $A \to B$ is a functional dependency and $B \to C$ is not a minimal functional dependency.

Functional dependencies can be calculated by means of partitions [6] as follows:

Property 2.2.2. Let $A, B \subseteq AT$. $A \rightarrow B$ is a functional dependency iff $\pi_A = \pi_{AB}$ iff $|\pi_A| = |\pi_{AB}|$.

Example 2.2.2. Let us consider the information system in Table 2. We observe that $\pi_{\{ce\}} = \pi_{\{cea\}} = \{\{1\},\{2\},\{3,4\},\{5\},\{6,7,8\},\{9,10\}\}$. The equality of $\pi_{\{ce\}}$ and $\pi_{\{cea\}}$ (or their cardinalities) is sufficient to conclude that $\{ce\} \rightarrow \{a\}$ is a functional dependency. □

The next property recalls a method of calculating a partition with respect to an attribute set C by intersecting partitions with respect to subsets of C. Let $A, B \subseteq AT$. The *product of partitions* π_A and π_B, denoted by $\pi_A \cap \pi_B$, is defined as $\pi_A \cap \pi_B = \{Y \cap Z | Y \in \pi_A \text{ and } Z \in \pi_B\}$.

Property 2.2.3. Let $A, B, C \subseteq AT$ and $C = A \cup B$. Then, $\pi_C = \pi_A \cap \pi_B$.

2.3 Decision Tables, Reducts and Functional Dependencies

A *decision table* is an information system $DT = (O, AT \cup \{d\})$, where $d \notin AT$ is a distinguished attribute called the *decision*, and the elements of AT are called *conditions*. A *decision class* is defined as the set of all objects with the same decision value. By X_{d_i} we will denote the decision class consisting of objects the decision value of which equals d_i, where $d_i \in V_d$. Clearly, for any object x in O, $I_d(x)$ is a decision class. It is often of interest to find minimal subsets of AT (or *strict reducts*) that functionally determine d. It may happen, nevertheless, that such minimal sets of conditional attributes do not exist.

Table 2. Sample decision table $DT = (O, AT \cup \{d\})$, where $AT = \{a, b, c, e, f\}$ and d is the decision attribute, extended with derived attributes d^{N}_{AT}, ∂_{AT}, μ_d^{AT}

oid	*a*	*b*	*c*	*e*	*f*	*d*	d^{N}_{AT}	∂_{AT}	$\mu_d^{AT} :< \mu_1^{AT}, \mu_2^{AT}, \mu_3^{AT} >$
1	1	0	0	1	1	1	1	$\{1\}$	$< 1, 0, 0 >$
2	1	1	1	1	2	1	1	$\{1\}$	$< 1, 0, 0 >$
3	0	1	1	0	3	1	N	$\{1, 2\}$	$< 1/2, 1/2, 0 >$
4	0	1	1	0	3	2	N	$\{1, 2\}$	$< 1/2, 1/2, 0 >$
5	0	1	1	2	2	2	2	$\{2\}$	$< 0, 1, 0 >$
6	1	1	0	2	2	2	N	$\{2, 3\}$	$< 0, 1/3, 2/3 >$
7	1	1	0	2	2	3	N	$\{2, 3\}$	$< 0, 1/3, 2/3 >$
8	1	1	0	2	2	3	N	$\{2, 3\}$	$< 0, 1/3, 2/3 >$
9	1	1	0	3	2	3	3	$\{3\}$	$< 0, 0, 1 >$
10	1	0	0	3	2	3	3	$\{3\}$	$< 0, 0, 1 >$

Example 2.3.1. Table 2 describes a sample decision table $DT = (O, AT \cup \{d\})$, where $AT = \{a, b, c, e, f\}$. Partition $\pi_{AT} = \{\{1\}, \{2\}, \{3,4\}, \{5\}, \{6,7,8\}, \{9\}, \{10\}\}$ contains all AT-indiscernibility classes, whereas $\pi_{\{d\}} = \{\{1,2,3\}, \{4,5,6\}, \{7,8,9,10\}\}$ contains all decision classes. There is no functional dependency between AT and d, since there is no decision class in $\pi_{\{d\}}$ containing AT-indiscernibility class $\{3,4\}$ (or $\{6,7,8\}$). As $AT \rightarrow d$ is not functional, then $C \rightarrow d$, where $C \subseteq AT$, is not functional either. □

Rough Sets theory deals with the problem of non-existence of strict reducts by means of other types of reducts, which always exist, irrespectively if $AT \rightarrow d$ is a functional dependency, or not. We will now recall such three types of reducts, namely certain decision reducts, generalized decision reducts, and membership distribution reducts.

Certain decision reducts. Certain decision reducts are defined based on the notion of a *positive region* of DT, thus we start with introducing this notion. A *positive region* of DT, denoted as POS, is the set-theoretical union of all AT-indiscernibility classes, each of which is contained in a decision class of DT; that is, $POS = \bigcup\{X \in \pi_{AT} | X \subseteq Y, Y \in \pi_d\} = \{x \in O | I_{AT}(x) \subseteq I_d(x)\}$. A set of attributes $A \subseteq AT$ is called a *certain decision reduct* of DT, if A is a minimal set, such that $\forall_{x \in POS}\ I_A(x) \subseteq I_d(x)$ [18]. Now, we will introduce a *derivable decision attribute* for an object $x \in O$ as a modification of the decision attribute d, which we will denote by $d^{\mathrm{N}}_{AT}(x)$ and define as follows: $d^{\mathrm{N}}_{AT}(x) = d(x)$ if $x \in POS$, and $d^{\mathrm{N}}_{AT}(x) = \mathrm{N}$, otherwise (see Table 2 for illustration). Clearly, all objects with values of d^{N}_{AT} that are different from N belong to POS.

Property 2.3.1 [10]**.** Let $A \subseteq AT$. A is a certain decision reduct iff $A \rightarrow \{d^{\mathrm{N}}_{AT}\}$ is a minimal functional dependency.

Generalized decision reducts. Generalized decision reducts are defined based on a *generalized decision.* Let us thus start with introducing this notion. An A-*generalized decision* for object x in DT (denoted by $\partial_A(x)$), $A \subseteq AT$, is defined as the set of all decision values of all objects indiscernible with x wrt. A; i.e., $\partial_A(x) = \{d(y) | y \in I_A(x)\}$ [21]. For $A = AT$, an A-*generalized decision* is also called a *generalized decision* (see Table 2 for illustration). $A \subseteq AT$ is defined a *generalized decision reduct* of DT, if A is a minimal set such that $\forall_{x \in O}$ $\partial_A(x) = \partial_{AT}(x)$.

Property 2.3.2 [10]**.** Let $A \subseteq AT$. Attribute set A is a generalized decision reduct iff $A \rightarrow \{\partial_{AT}\}$ is a minimal functional dependency.

μ-Decision Reducts. The generalized decision informs on decision classes to which an object may belong, but does not inform on the degree of the membership to these classes, which could be also of interest. A *membership distribution function*) $\mu^A_d : O \rightarrow [0,1]^n, A \subseteq AT, n = |V_d|$, is defined as follows [9],[23-24]:

$$\mu^A_d(x) = (\mu^A_{d_1}(x), \ldots, \mu^A_{d_n}(x)), \text{ where}$$
$$\{d_1, \ldots, d_n\} = V_d \text{ and } \mu^A_{d_i}(x) = \frac{|I_A(x) \cap X_{d_i}|}{|I_A(x)|}.$$

Please, see Table 2 for illustration of μ^{AT}_d. $A \subseteq AT$ is a called a μ-*decision reduct* (or *membership distribution reduct*) of DT, if A is a minimal set such that $\forall_{x \in O}\ \mu^A_d(x) = \mu^{AT}_d(x)$.

Property 2.3.3 [10]**.** Let $A \subseteq AT$. A is a μ-decision reduct iff $A \rightarrow \{\mu^{AT}_d\}$ is a minimal functional dependency.

3 Computing Minimal Sets of Attributes Functionally Determining Given Dependent Attribute with Fun

In this section, we present the *Fun* algorithm for computing all minimal subsets of conditional attributes AT that functionally determine a given dependent attribute ∂. First, we recall the variants of Fun that apply partitions of objects or, so called, stripped partitions of objects [12]. Then, in Section 3.5, we introduce an idea of reduced stripped partitions and offer two new variants of *Fun* based on them.

The *Fun* algorithm can be used for calculating Rough Sets reducts provided the dependent attribute is determined properly, namely *Fun* will return certain decision reducts for $\partial = \partial_{AT}$, generalized decision for $\partial = d^{\mathrm{N}}_{AT}$, and μ-decision reducts for $\partial = \mu^{AT}_{d}$. For brevity, a minimal subset of AT that functionally determines a given dependent attribute ∂ will be called a ∂-*reduct*.

3.1 Main Algorithm

The *Fun* algorithm takes two arguments: a set of conditional attributes AT and a functionally dependent attribute ∂. As a result, it returns all ∂-reducts. *Fun* starts with creating singleton candidates $\mathcal{C}_1$ for ∂-reducts from each attribute in AT. Then, the partitions (π) and their cardinalities (*groupNo*) wrt. ∂ and all attributes in $\mathcal{C}_1$ are determined.

Notation for Fun	
• $\mathcal{C}_k$	candidate k attribute sets (potential ∂-reducts);
• $\mathcal{R}_k$	k attribute ∂-reducts;
• $C.\pi$	the representation of the partition π_C of the candidate attribute set C; it is stored as the list of groups of objects identifiers (*oids*);
• $C.groupNo$	the number of groups in the partion of the candidate attribute set C; that is, $\lvert\pi_C\rvert$;
• $\partial.T$	an array representation of π_∂;

```
Algorithm. $Fun$(attribute set $AT$, dependent attribute $\partial$);
$\mathcal{C}_1 = \{\{a\}|a \in AT\}$;          // create singleton candidates from conditional attributes in $AT$
forall $C$ in $\mathcal{C}_1 \cup \{\partial\}$ do begin
  $C.\pi = \pi_C$;
  $C.groupNo = |\pi_C|$
endfor;
/* calculate an array representation of $\pi_\partial$ for later multiple use in the Holds function */
$\partial.T = PartitionArrayRepresentation(\partial)$;
for ($k = 1$; $\mathcal{C}_k \neq \emptyset$; $k++$) do begin                              // Main loop
  $\mathcal{R}_k = \{\}$;
  forall candidates $C \in \mathcal{C}_k$ do begin
    if $Holds(C \rightarrow \{\partial\})$ then                 // Is $C \rightarrow \{\partial\}$ a functional dependency?
      remove $C$ from $\mathcal{C}_k$ to $\mathcal{R}_k$;                  // store $C$ as a $k$ attribute $\partial$-reduct
    endif
  endfor;
  /* create $(k+1)$ attribute candidates for $\partial$-reducts from $k$ attribute non-$\partial$-reducts */
  $\mathcal{C}_{k+1} = FunGen(\mathcal{C}_k)$;
endfor;
return $\bigcup_k \mathcal{R}_k$;
```

Next, the *PartitionArrayRepresentation* function (see Section 3.3) is called to create an array representation of π_∂. This representation shall be used multiple times in the *Holds* function, called later in the algorithm, for efficient checking whether candidate attribute sets determine ∂ functionally. Now, the main loop starts. In each k-th iteration, the following is performed:

– The *Holds* function (see Section 3.3) is called to check if k attribute candidates $\mathcal{C}_k$ determine ∂ functionally. The candidates that do are removed from the set of k attribute candidates to the set of ∂-reducts R_k.
– The *FunGen* function (see Section 3.2) is called to create $(k+1)$ attribute candidates $\mathcal{C}_{k+1}$ from the k attribute candidates that remained in $\mathcal{C}_k$.

The algorithm stops when the set of candidates becomes empty.

3.2 Generating Candidates for ∂-reducts

The *FunGen* function creates $(k+1)$ attribute candidates $\mathcal{C}_{k+1}$ by merging k attribute candidates $\mathcal{C}_k$, which are not ∂-reducts. The algorithm adopts the manner of creating and pruning of candidates introduced in [1] (here: candidate sets of attributes instead of candidates for frequent itemsets). There are merged only those pairs of k attribute candidates $\mathcal{C}_k$ that differ merely on their last attributes (see [1] for justification that this method is lossless and non-redundant). For each new candidate C, π_C is calculated as the product of the partitions wrt. the merged k attribute sets (see Section 3.3 for the *Product* function). The cardinality (*groupNo*) of π_C is also calculated. Now, it is checked for each new $(k+1)$ attribute candidate C, if there exists its k attribute subset A not present in $\mathcal{C}_k$. If so, it means that either A or its subset was found earlier as a ∂-reduct. This implies that the candidate C is a proper superset of a ∂-reduct, thus it is not a ∂-reduct, and hence C is deleted from the set $\mathcal{C}_{k+1}$. Optionally, for each tested k attribute subset A that is present in $\mathcal{C}_k$, it is checked, if $|\pi_A|$ equals $|\pi_C|$. If so, then $A \rightarrow C$ holds (by Property 2.2.2). Hence, $A \rightarrow \{\partial\}$ is not a minimal functional dependency (by Property 2.2.1e), and thus C is deleted from $\mathcal{C}_{k+1}$.

```
function $FunGen(\mathcal{C}_k)$;
/* Merging */
forall $A, B \in \mathcal{C}_k$ do
  if $A[1] = B[1] \wedge \ldots \wedge A[k-1] = A[k-1] \wedge A[k] < B[k]$ then begin
    $C = A[1] \cdot A[2] \cdot \ldots \cdot A[k] \cdot B[k]$;
    /* compute partition $C.\pi$ as a product of $A.\pi$ and $B.\pi$, and the number of groups in $C.\pi$ */
    $C.groupNo = Product(A.\pi, B.\pi, C.\pi)$;
    add $C$ to $\mathcal{C}_{k+1}$
  endif;
endfor;
/* Pruning */
forall $C \in \mathcal{C}_{k+1}$ do
  forall $k$ attribute set $A$, such that $A \subset C$ do
    if $A \notin \mathcal{C}_k$ then
      /* $A \subset C$ and $\exists B \subseteq A$ such that $B \rightarrow \{\partial\}$ holds, so $C \rightarrow \partial$ holds, but is not minimal */
      begin delete $C$ from $\mathcal{C}_{k+1}$; break
      end
    elseif $A.groupNo = C.groupNo$ then                    // optional candidate pruning step
      /*$A \subset C$ and $A \rightarrow C$ holds, so $C \rightarrow \{\partial\}$ is not a minimal functional dependency */
      begin delete $C$ from $\mathcal{C}_{k+1}$; break
      end
    endif
  endfor
endfor;
return $\mathcal{C}_{k+1}$;
```

3.3 Using Partitions in Fun

Computing Array Representation of Partition. The *PartitionArrayRepresentation* function returns an array T of the length equal to the number of objects O in DT. For a given attribute C, each element j in T is assigned the index of the group in $C.\pi$ to which object with $oid = j$ belongs. As a result, j-th element of T informs to which group in $C.\pi$ j-th object in DT belongs, $j = 1..\,|O|$.

```
function $PartitionArrayRepresentation$(attribute set $C$);
/* assert: $T$ is an array[$1 \ldots |O|$] */
$i = 1$;
for $i$-th group $G$ in partition $C.\pi$ do begin
   for each $oid$ $G$ do $T[oid] = i$ endfor;
   $i = i + 1$
endfor
return $T$;
```

Verifying Candidate Dependency. The *Holds* function checks, if there is a functional dependency between the set of attributes C and an attribute ∂. It is checked for successive groups G in $C.\pi$, if there is an *oid* in G that belongs to a group in $\partial.\pi$ different from the group in $\partial.\pi$ to which the first *oid* in G belongs (for the purpose of efficiency, the pre-calculated $\partial.T$ representation of the partition for ∂ is applied instead of $\partial.\pi$). If so, this means that G is not contained in one group of $\partial.\pi$ and thus $C \rightarrow \{\partial\}$ is not a functional dependency. In such a case, the function stops returning **false** as a result. Otherwise, if no such group G is found, the function returns **true**, which means that $C \rightarrow \{\partial\}$ is a functional dependency.

```
function $Holds(C \rightarrow \{\partial\})$;
/* assert: $\partial.T$ is an array representation of $\partial.\pi$ */
for each group $G$ in partition $C.\pi$ do begin
   $oid$ = first element in group $G$;
   $\partial$-$firstGroup = \partial.T[oid]$;       // the identifier of the group in $\partial.\pi$ to which $oid$ belongs
   for each next element $oid \in G$ do begin
      $\partial$-$nextGroup = \partial.T[oid]$;
      if $\partial$-$firstGroup \neq \partial$-$nextGroup$ then
         /* there are $oids$ in $G$ that identify objects indiscernible wrt. $C$, but discernible wrt. $\partial$ */
         return false                                   // hence, $C \rightarrow \{\partial\}$ does not hold
      endif
   endfor;
endfor;
return true;                                                    // $C \rightarrow \{\partial\}$ holds
```

Computing Product of Partitions. The *Product* function computes the partition wrt. the attribute set C and its cardinality from the partitions wrt. the attribute sets A and B. The function examines successive groups wrt. the partition for B. The objects in a given group G in $B.\pi$ are split into maximal subgroups in such a way that the objects in each resultant subgroup are contained in a same group in $A.\pi$. The obtained set of subgroups equals $\{G \cap Y | Y \in A.\pi\}$. Product $C.\pi$ is calculated as the set of all subgroups obtained from all groups in $B.\pi$; i.e., $C.\pi = \bigcup_{G \in B.\pi} \{G \cap Y | Y \in A.\pi\} = \{G \cap Y | Y \in A.\pi \text{ and } G \in B.\pi\} = B.\pi \cap A.\pi$. In order to calculate the product of the partitions efficiently (with time complexity linear wrt. the number of objects in DT), we follow the idea presented in [6], and use two static arrays T and S: T is used to store an array representation of the partition wrt. A; S is used to store subgroups obtained from a given group G in $B.\pi$.

```
function $Product(A.\pi, B.\pi;$ var $C.\pi)$;
/* assert: $T[1..|O|]$ is a static array */
/* assert: $S[1..|O|]$ is a static array with all elements initially equal to $\emptyset$ */
$C.\pi = \{\}$; $groupNo = 0$;
/* calculate an array representation of $A.\pi$ for later multiple use in the $Product$ function */
$T = PartitionArrayRepresentation(A)$; $i = 1$;
for $i$-th group $G$ in partition $B.\pi$ do begin
  $A\text{-}GroupIds = \emptyset$;
  for each element $oid \in G$ do begin
    $j = T[oid]$;            // the identifier of the group in $A.\pi$ to which $oid$ belongs
    insert $oid$ into $S[j]$; insert $j$ into $A\text{-}GroupIds$
  endfor;
  for each $j \in A\text{-}GroupIds$ do begin
    insert $S[j]$ into $C.\pi$;
    $groupNo = groupNo + 1$; $S[j] = \emptyset$
  endfor;
  $i = i + 1$
endfor;
return $groupNo$;
```

3.4 Using Stripped Partitions in Fun

The representation of partitions that requires storing objects identifiers (*oids*) of all objects in *DT* may be too memory consuming. In order to alleviate this problem, it was proposed in [6] to store *oids* only for objects belonging to non-singleton groups in a partition representation. Such a representation of a partition is called a *stripped* one and will be denoted by π^s. Clearly, the stripped representation is lossless.

Example 3.4.1. In Table 2, the partition wrt. $\{ce\}$: $\pi_{\{ce\}} = \{\{1\}, \{2\}, \{3,4\}, \{5\}, \{6,7,8\}, \{9,10\}\}$, whereas the stripped partition wrt. $\{ce\}$: $\pi^s_{\{ce\}} = \{\{3,4\}, \{6,7,8\}, \{9,10\}\}$. □

```
function $StrippedHolds(C \rightarrow \{\partial\})$;
$i = 1$;
for $i$-th group $G$ in partition $C.\pi$ do begin
  $oid$ = first element in group $G$;
  $\partial\text{-}firstGroup = \partial.T[oid]$;     // the identifier of the group in $\partial.\pi$ to which $oid$ belongs
  if $\partial\text{-}firstGroup$ = null then return false endif;
  /* $\partial.T[oid]$ = null indicates that $oid$ constitutes a singleton group in the partition for $\partial$. */
  /* Hence, no next object in $G$ belongs to this group in $\partial.\pi$ , so $C \rightarrow \{\partial\}$ does not hold.  */
  for each next element $oid \in G$ do begin
    $\partial\text{-}nextGroup = \partial.T[oid]$;
    if $\partial\text{-}firstGroup \neq \partial\text{-}nextGroup$ then
      /* there are $oids$ in $G$ that identify objects indiscernible wrt. $C$, but discernible wrt. $\partial$ */
      return false                       // hence, $C \rightarrow \{\partial\}$ does not hold
    endif
  endfor;
  $i = i + 1$
endfor;
return true;                             // $C \rightarrow \{\partial\}$ holds
```

When applying stripped partitions in our *Fun* algorithm instead of usual partitions, one should call the *StrippedHolds* function instead of *Holds*, and the *StrippedProduct* function instead of *Product*. The modified parts of the functions have been shadowed in the code below. We note, however, that the *groupNo* field still stores the number of groups in an unstripped partition (singleton groups are not stored in a stripped partition, but are counted!).

```
function $StrippedProduct(A.\pi, B.\pi;$ var $C.\pi)$;
$C.\pi = \{\}$; $groupNo = B.groupNo$;
$T = PartitionArrayRepresentation(A)$; $i = 1$;
for $i$-th group $G$ in partition $B.\pi$ do begin
  $A-GroupIds = \emptyset$;
  for each element $oid \in G$ do begin
    $j = T[oid]$;                  // the identifier of the group in $A.\pi$ to which $oid$ belongs
    if $j =$ null then $groupNo = groupNo + 1$;           // respect singleton subgroups
    else begin insert $oid$ into $S[j]$; insert $j$ into $A$-$GroupIds$ endif
  endfor;
  for each $j \in A-GroupIds$ do begin
    if $|S[j]| > 1$ then
      insert $S[j]$ into $C.\pi$                          // store only non-singleton groups
    endif ;
    $groupNo = groupNo + 1$; $S[j] = \emptyset$      // but count all groups, including singleton ones
  endfor;
  $groupNo = groupNo - 1$;
  $i = i + 1$
endfor;
/* Clearing of array $T$ for later use */
for $i$-th group $G$ in partition $A.\pi$ do
  for each element $oid \in G$ do $T[oid] =$ null endfor
endfor;
return $groupNo$;
```

3.5 Using Reduced Stripped Partitions in Fun

In this section, we offer further reduction of stripped partitions wrt. conditional attributes. Our proposal is based on the following observations:

Let C be a conditional attribute set and d be the decision attribute. Let G be any group in the stripped partition wrt. C that is contained in a group belonging to the stripped partition wrt. d.

a) Group G operates in favour of functional dependency between C and d.
b) Any subgroup $G' \subseteq G$ that occurs in the stripped partition wrt. a superset $C' \supseteq C$ also operates in favour of functional dependency between C' and d. Thus, the verification of the containment of G' in a group of the stripped partition wrt. d is dispensable in testing the existence of a functional dependency between C' and d.

We define a *reduced stripped partition wrt. attribute set* A (and denote by π^{rs}_A) as the set of those groups in the stripped partition wrt. A that are not contained in any group in the stripped partition wrt. decision d; that is, $\pi^{rs}_A = \{G \in \pi^s_A | \neg\exists_{D\in\pi^s_{\{d\}}} G \subseteq D\}$.

Example 3.5.1. In Table 2, the stripped partition wrt. conditional attribute e: $\pi^s_{\{e\}} = \{\{1,2\}, \{3,4\}, \{5,6,7,8\}, \{9,10\}\}$, whereas the stripped partition wrt. decision attribute d: $\pi^s_{\{d\}} = \{\{1,2,3\}, \{4,5,6\}, \{7,8,9,10\}\}$. We note that group

$\{1,2\} \in \pi^s_{\{e\}}$ and its subsets are contained in group $\{1,2,3\} \in \pi^s_{\{d\}}$. Similarly, group $\{9,10\} \in \pi^s_{\{e\}}$ and its subsets are contained in group $\{7,8,9,10\} \in \pi^s_{\{d\}}$. There is no group in $\pi^s_{\{d\}}$ containing $\{3,4\}$ or $\{5,6,7,8\}$. Thus, the groups $\{1,2\}$ and $\{9,10\}$ in $\pi^s_{\{e\}}$, unlike the remaining two groups: $\{3,4\}$ and $\{5,6,7,8\}$ in $\pi^s_{\{e\}}$, operate in favour of functional dependency between $\{e\}$ and $\{d\}$. Hence, the reduced stripped partition $\pi^s_{\{e\}} = \{\{3,4\}, \{5,6,7,8\}\}$ and the reduced stripped partitions wrt. supersets of $\{e\}$ will contain neither $\{1,2\}$, nor $\{9,10\}$, nor their subsets. □

It is easy to observe that the reduced stripped partition wrt. attribute set C can be calculated based on the product of reduced stripped partitions wrt. subsets of C as shown in Proposition 3.5.1.

Proposition 3.5.1. Let $A, B, C \subseteq AT$ and $C = A \cup B$. Then, the reduced stripped partition wrt. C equals the set of the groups in the product of the reduced stripped partitions wrt. A and B that are not contained in any group of the stripped partition wrt. decision d; that is, $\pi^{rs}_C = \{G \in \pi^{rs}_A \cap \pi^{rs}_B \mid \neg\exists_{D \in \pi^s_{\{d\}}} G \subseteq D\}$.

```
function ReducedStrippedHolds($C \rightarrow \{\partial\}$);
$i = 1$;
holds = true;
for $i$-th group $G$ in partition $C.\pi$ do begin
  oid = first element in group $G$;
  $\partial$-firstGroup = $\partial.T[oid]$;          // the identifier of the group in $\partial.\pi$ to which oid belongs
  if $\partial$-firstGroup = null then holds = false;
  /* $\partial.T[oid]$ = null indicates that oid constitutes a singleton group in the partition for $\partial$. */
  /* Hence, no next object in $G$ belongs to this group in $\partial.\pi$ , so $C \rightarrow \{\partial\}$ does not hold.   */
  else begin
    for each next element oid $\in G$ do begin
      $\partial$-nextGroup = $\partial.T[oid]$;
      if $\partial$-firstGroup $\neq$ $\partial$-nextGroup then
        /* there are oids in $G$ that identify objects indiscernible wrt. $C$, but discernible wrt. $\partial$ */
        holds = false; break;                        // hence, $C \rightarrow \{\partial\}$ does not hold
      endif
    endfor;
    if $\partial$-firstGroup $\neq$ $\partial$-nextGroup then
      delete $c$ from $C.\pi$;
    endif;
    $i = i + 1$;
  endif;
endfor;
return holds ;
```

In our proposal, the product $\pi^{rs}_A \cap \pi^{rs}_B$ of the reduced stripped partitions wrt. A and B is calculated by means of the *StrippedProduct* function. The reduced stripped partition π^{rs}_C is determined from a product $\pi^{rs}_A \cap \pi^{rs}_B$ by new *ReducedStrippedHolds* function. The function is a modification of *StrippedProduct*. The *ReducedStrippedHolds* function like *StrippedProduct* verifies, if there is a functional dependency between C and d. In addition, *ReducedStrippedHolds* removes those groups in product $\pi^{rs}_A \cap \pi^{rs}_B$ that are contained in $\pi^s_{\{d\}}$. The modified parts of the code in *ReducedStrippedHolds* have been shadowed.

Please, note that the *StrippedHolds* function reads groups of $\pi^s_C = \pi^s_A \cap \pi^s_B$ until the first group that is not contained in a group of $\pi^s_{\{d\}}$ is found. To the contrary, *ReducedStrippedHolds* reads all groups of the product $\pi^{rs}_A \cap \pi^{rs}_B$. This means that the execution of *ReducedStrippedHolds* may last longer than the execution of *StrippedHolds*, when $\pi^{rs}_A \cap \pi^{rs}_B$ and $\pi^s_A \cap \pi^s_B$ are of similar length. On the other hand, the execution of *ReducedStrippedHolds* may last shorter than the execution of *StrippedHolds*, when $\pi^{rs}_A \cap \pi^{rs}_B$ is shorter than $\pi^s_A \cap \pi^s_B$. As an alternative to both solutions of shortening partitions, we propose the *PartReducedStrippedHolds* function, which deletes the groups from the product $\pi^s_A \cap \pi^s_B$ until the first group in this product that is not contained in a group of $\pi^s_{\{d\}}$ is found. The result of *PartReducedStrippedHolds* is a group set being a subset of $\pi^s_A \cap \pi^s_B$ and superset of $\pi^{rs}_A \cap \pi^{rs}_B$.

```
function $PartReducedStrippedHolds(C \rightarrow \{\partial\})$;
$i = 1$;
for $i$-th group $G$ in partition $C.\pi$ do begin
  $oid$ = first element in group $G$;
  $\partial$-$firstGroup = \partial.T[oid]$;          // the identifier of the group in $\partial.\pi$ to which $oid$ belongs
  if $\partial$-$firstGroup$ = null then return false endif;
  /* $\partial.T[oid]$ = null indicates that $oid$ constitutes a singleton group in the partition for $\partial$. */
  /* Hence, no next object in $G$ belongs to this group in $\partial.\pi$ , so $C \rightarrow \{\partial\}$ does not hold.   */
  for each next element $oid \in G$ do begin
    $\partial$-$nextGroup = \partial.T[oid]$;
    if $\partial$-$firstGroup \neq \partial$-$nextGroup$ then
      /* there are $oids$ in $G$ that identify objects indiscernible wrt. $C$, but discernible wrt. $\partial$ */
      return false                                   // hence, $C \rightarrow \{\partial\}$ does not hold
    endif
  endfor;
  delete $c$ from $C.\pi$;
  $i = i + 1$;
endfor;
return true;                                          // $C \rightarrow \{\partial\}$ holds
```

We note that it is impossible to determine the number of groups in the product $\pi_A \cap \pi_B$ as a side-effect of calculating the product of the reduced stripped partitions $\pi^{rs}_A \cap \pi^{rs}_B$. The same observation holds when the product is calculated from the partially reduced stripped partitions. Lack of this knowledge disallows using the optional pruning step in the *FunGen* algorithm. The usefulness of using fully or partially reduced stripped partitions will be examined experimentally in Section 4.

4 Experimental Results

We have performed a number of experiments on a few data sets available in UCI Repository (`http://www.ics.uci.edu/~mlearn/MLRepository.html`) and

Table 3. Six variants of the *Fun* algorithm wrt. *Holds* algorithm, partitions' type and candidate pruning option

Fun's variant	H	H/P	SH	SH/P	PRSH	RSH
Holds method	Holds	Holds	Stripped Holds	Stripped Holds	Part Reduced Stripped Holds	Reduced Stripped Holds
Stripped partitions	No	No	Yes	Yes	Yes	Yes
Optional candidate pruning	No	Yes	No	Yes	No	No

Table 4. Reference external tools

Label	*Tane*	*RSES*	RSGR	RRER
Tool	Tane	RSES	Rosetta	Rosetta
Algorithm	Tane	Exhaustive	SAV Genetic Reducer	RSES Exhaustive Reducer
Comments				Limitation to 500 records

Table 5. Execution times in seconds for the letter-recognition data set. * - results are not available, a data set was to large to be analyzed. ** - ** - RSES was written in *Java*, which could cause an additional overhead, *** - the times provided by Rosetta have 1 second granularity.

	$\|O\|$	H	H/P	SH	SH/P	PRSH	RSH	RSES**	*Tane*	RSGR***	RRER***
1	100	0.32	0.52	0.36	0.23	0.30	0.24	9.50	0.55	<500 (or 0 sec)	14.00
2	200	2.54	2.72	0.87	0.25	0.97	0.97	13.50	0.74	1.00	5.00
3	500	7.92	7.24	1.63	0.79	1.71	1.57	9.00	1.56	1.00	28.00
4	1000	26.70	19.41	3.72	2.03	3.28	2.94	14.00	3.16	1.00	N/A*
5	2000	38.29	27.60	7.97	4.28	7.99	6.74	20.00	6.94	6.00	N/A*
6	5000	126.19	76.48	28.52	19.54	28.66	28.91	130.00	24.23	35.00	N/A*
7	10000	1687.15	976.04	51.51	52.97	59.97	131.79	960.00	73.07	180.00	N/A*
8	15000	N/A*	N/A*	154.79	137.21	144.86	368.67	1860.00	152.20	570.00	N/A*
9	20000	N/A*	N/A*	444.70	421.89	440.31	727.69	3060.00	822.81	1200.00	N/A*

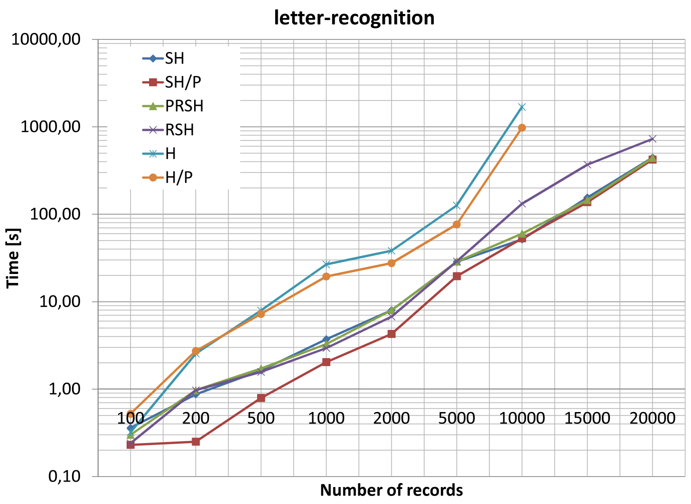


Fig. 1. Letter-recognition - logarithmic scale

other used by the Rough Sets community. In particular, we tested the following data sets: letter-recognition (16 conditional attributes; 20000 objects), diabetic (12 conditional attributes; 99 objects), krkopt (6 conditional attributes; 25000 objects), and nursary (9 conditional attributes; 11500 objects). The experiments we performed cover six different variants of the *Fun* algorithm and report the execution times for different size data samples for each tested data set. In this section, we apply brief names for these variants, as presented in Table 3.

Table 6. Execution times in miliseconds for the diabetic data set. * - the times provided by *Rosetta* have 1 second granularity.

	$\|O\|$	H	H/P	SH	SH/P	PRSH	RSH	RSES	*Tane*	RSGR*	RRER*
1	11	20	10	10	0	10	10	<500 (or 0 sec)	10	<500 (or 0 sec)	<500 (or 0 sec)
2	22	30	10	10	0	10	10	<500 (or 0 sec)	10	<500 (or 0 sec)	<500 (or 0 sec)
3	33	41	10	10	0	10	10	<500 (or 0 sec)	20	<500 (or 0 sec)	<500 (or 0 sec)
4	44	20	20	20	0	20	20	<500 (or 0 sec)	20	<500 (or 0 sec)	<500 (or 0 sec)
5	55	30	20	20	0	10	20	<500 (or 0 sec)	29	<500 (or 0 sec)	<500 (or 0 sec)
6	66	30	30	20	10	20	20	<500 (or 0 sec)	20	<500 (or 0 sec)	<500 (or 0 sec)
7	77	30	30	20	10	20	30	<500 (or 0 sec)	20	<500 (or 0 sec)	<500 (or 0 sec)
8	88	40	40	30	10	20	30	<500 (or 0 sec)	20	<500 (or 0 sec)	<500 (or 0 sec)
9	99	40	40	30	10	30	30	<500 (or 0 sec)	30	<500 (or 0 sec)	<500 (or 0 sec)

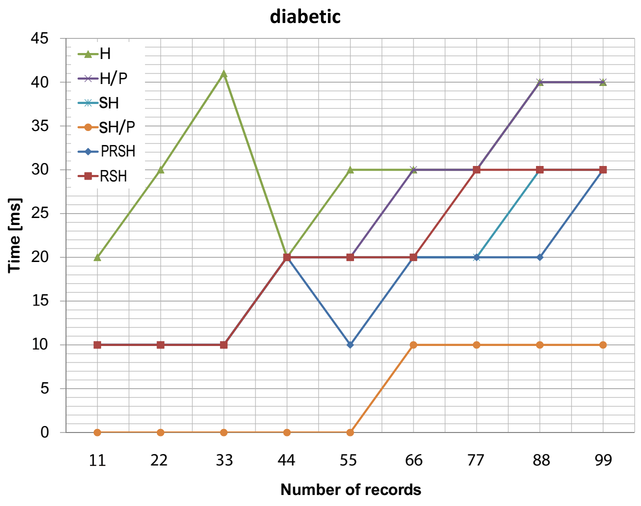


Fig. 2. Diabetic - linear scale

Table 7. Execution times in miliseconds for the krkopt data set. * - results are not available, a data set was to large to be analyzed ** - *RSES* was written in *Java*, which could cause an additional overhead, *** - the times provided by *Rosetta* have 1 second granularity.

	$\|O\|$	H	H/P	SH	SH/P	PRSH	RSH	RSES**	*Tane*	RSGR***	RRER***
1	2000	10	10	10	10	10	10	7800	40	2000	N/A*
2	5000	40	40	40	40	50	130	45000	60	16000	N/A*
3	10000	80	90	80	90	80	391	240000	130	75000	N/A*
4	15000	140	140	130	141	131	671	600000	200	227000	N/A*
5	20000	170	181	180	170	190	941	1290000	270	493000	N/A*
6	25000	240	220	230	241	220	1232	2460000	350	305000	N/A*

In addition to comparing the performance of the variants of *Fun*, we tested the performance of *Fun* against the performance of three external tools: RSES and Rosetta (both available in the software repository at http://rsds.univ.rzeszow.pl) and Tane (available at http://www.cs.helsinki.fi/research/fdk/datamining/tane). The list of these the tools is shown in Table 4.

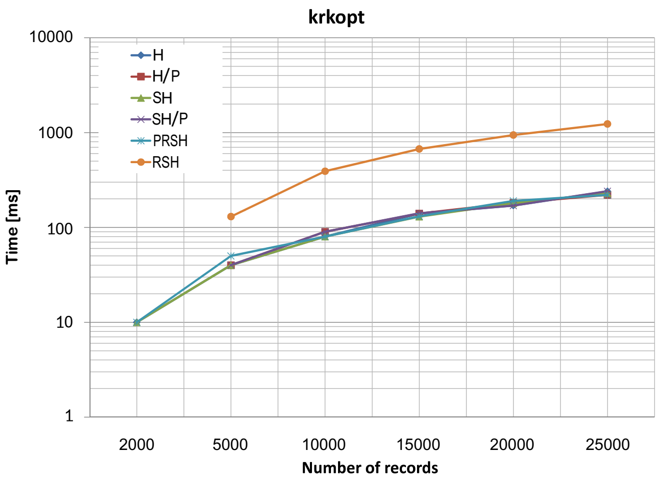


Fig. 3. Krkopt - logarithmic scale

Table 8. Execution times in miliseconds for the nursery data set. * - results are not available, a data set was to large to be analyzed ** - *RSES* was written in *Java*, which could cause an additional overhead, *** - the times provided by *Rosetta* have 1 second granularity.

	$\|O\|$	H	H/P	SH	SH/P	PRSH	RSH	RSES**	*Tane*	RSGR***	RRER***
1	4500	141	141	150	151	150	150	46000	370	15000	N/A*
2	6000	200	201	200	191	200	210	120000	321	27000	N/A*
3	7500	240	240	250	240	260	290	240000	410	41000	N/A*
4	9000	290	301	300	301	320	360	390000	460	74000	N/A*
5	10500	351	340	351	360	360	470	540000	540	113000	N/A*

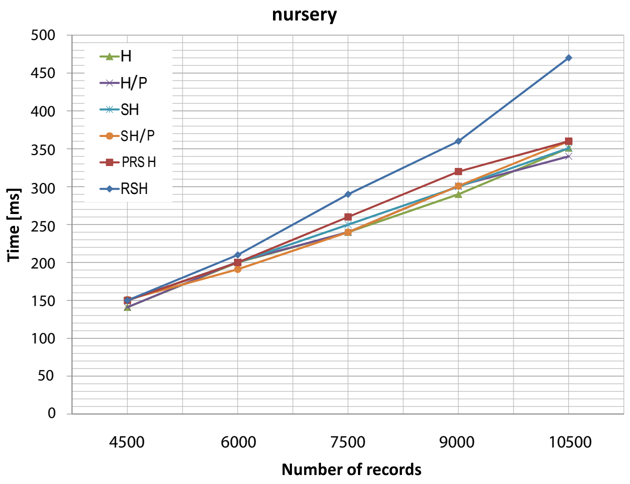


Fig. 4. Nursery - linear scale

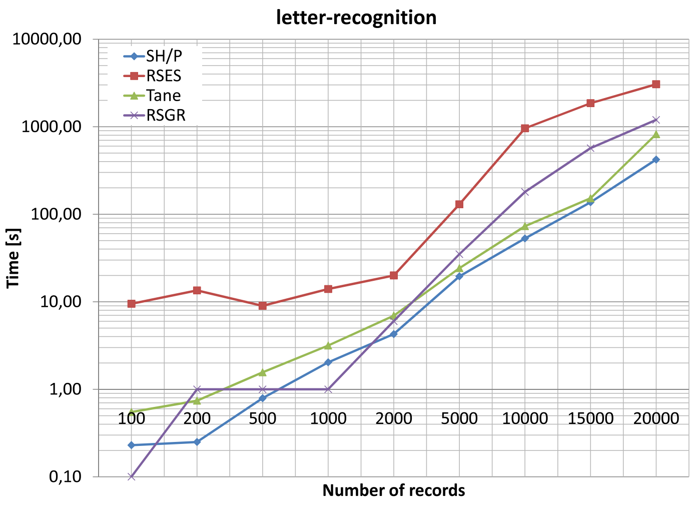


Fig. 5. Letter-recognition - logarithmic scale

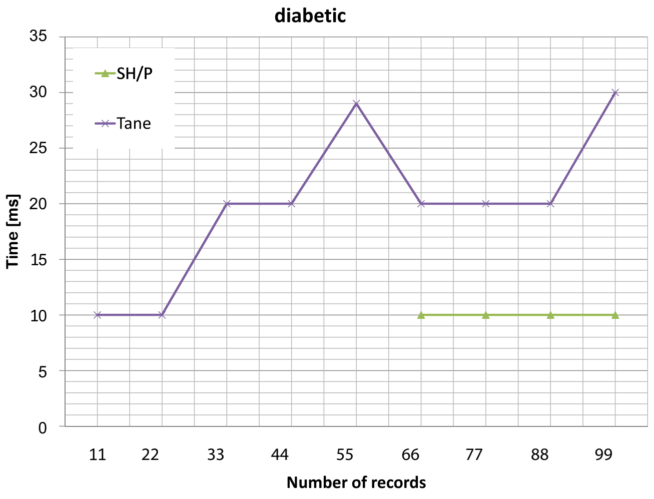


Fig. 6. Diabetic - linear scale

It follows from the obtained results, that there is no one winner among the variants of *Fun*. In the case of the letter-recognition data set (see Table 5 and Figure 1), there are three leading variants, namely SH/P, PRSH and SH. RSH shows similar behaviour for up to 5000 objects and then becomes slower up to two times in comparison with the leaders. In the case of large number of objects, H is up to 33 times slower than the leading three variants and H/P is up to 19 times slower than the leading three variants. The observed behaviour suggests that (reduced) stripped partitions are much concise than the original partitions.

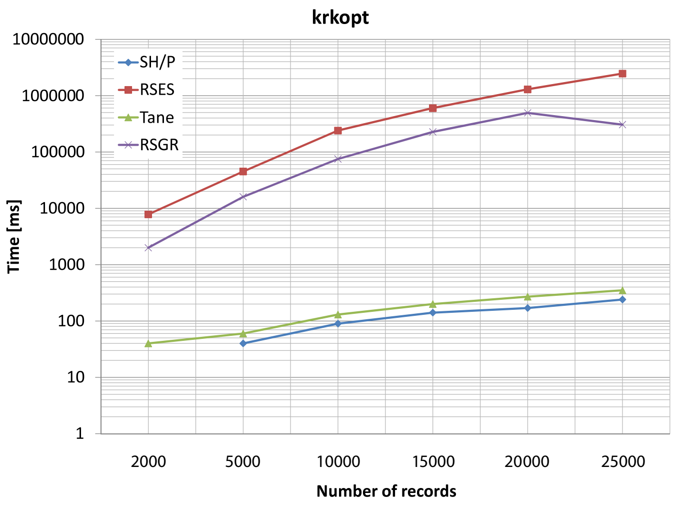


Fig. 7. Krkopt - logarithmic scale

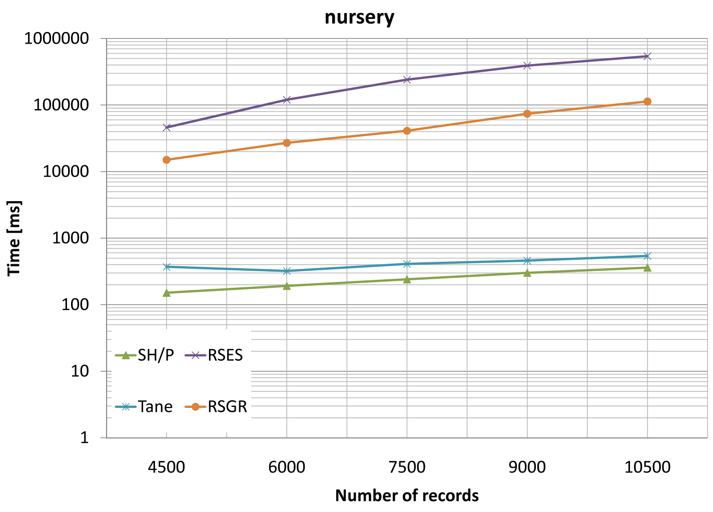


Fig. 8. Nursery - linear scale

In addition, the optional pruning must decrease the number of candidate dependencies considerably.

For the diabetic data set, the winner is SH/P (see Table 6 and Figure 2). The other variants using stripped partitions, namely: PRSH, SH, and RSH lasted 2 to 3 times longer. The variants that do not use stripped partitions; that is, H and H/P turned out slower than SH/P up to 4 times. The candidate pruning speeded up the performance of *Fun* mainly in the case of applying stripped partitions.

In the case of the *krkopt* data set, 5 variants of *Fun* show similar behaviour (see Table 7 and Figure 3). The worst variant is RSH, which starts to be slower than the other variants for the data sample containg 5000 objects or more. In the worst reported case, RSH lasted around 5 times longer than the other variants. This suggests that the full reduction of stripped partitions, which may be quite time consuming, did not lead to the real removal of the partitions' groups.

In the case of the nursery data set, two variants H and H/P using only the unstripped partitions turned out slightly faster than the SH/P, PRSH and SH variants, which in turn turned out slightly faster than RSH (see Table 8 and Figure 4).

Although there is no one winner among the variants of *Fun*, we note that for the datasets with larger number of conditional attributes: letter-recognition and diabetic, SH/P is the fastest variant of *Fun*, while in the case of the other two datasets, there is no significant difference in performance of variants using unstripped partitions and the ones using reduced/stripped partitions.

Our experiments prove that *Fun* is faster than RSES, *Tane* and Rosetta for all large data sets: letter-recognition, krkopt, and nursary (see Tables 5, 7–8 and Figures 5, 7-8). The leading variants of *Fun* calculated functional dependencies significantly faster. In comparison with RSES and Rosetta, *Fun* was faster even by 3 orders of magnitude. We were not able to determine the relationship between time performance of *Fun* and RSES as well as Rosetta for small diabetic data set, since the calculations took miliseconds; that is, required measuring with the granularity not available in the tools. However, also for this dataset, the fastest *Fun*'s variant - SH/P was faster than *Tane* up to 30 times (see Table 6 and Figure 6).

5 Conclusions and Future Work

We have described our *Fun* algorithm for discovering minimal sets of conditional attributes functionally determining a decision attribute, and in particular for computing certain, generalized decision, and μ-distribution reducts. We have also offered how to reduce stripped partitions that are used to determine minimal functional dependencies for a given dependent attribute or Rough Set reducts. We have proposed two variants of *Fun* based on reduced stripped partitions. We have carried out a number of experiments on benchmark data sets. Although no variant of *Fun* turned out the only winner for all data sets, we noted that for the datasets with large number of conditional attributes (letter-recognition and diabetic), the variants of *Fun* using stripped and partially reduced stripped partitions were considerably faster (up to 33 times) than the variants using unstripped partitions. We need to verify this phenomenon for larger number of data sets. The experiments also show that *Fun* is consistently faster than *Tane*, which computes all minimal functional dependencies, and is up to 3 orders of magnitude faster than *SAVGeneticReducer* and *RSESExhaustiveReducer* from Rosetta as well as RSES.

Acknowledgements. This research has been supported by grant No 3 T11C 002 29 received from Polish Ministry of Education and Science.

References

1. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast Discovery of Association Rules. In: Advances in KDD, pp. 307–328. AAAI, Menlo Park (1996)
2. Armstrong, W.W.: Dependency Structures of Data Based Relationships. In: Proceedings of IFIP Congress, Geneva, Switzerland, pp. 580–583 (1974)
3. Bazan, J., Skowron, A., Synak, P.: Dynamic Reducts as a Tool for Extracting Laws from Decision Tables. In: Raś, Z.W., Zemankova, M. (eds.) ISMIS 1994. LNCS (LNAI), vol. 869, pp. 346–355. Springer, Heidelberg (1994)
4. Bazan, J., Nguyen, H.S., Nguyen, S.H., Synak, P., Wroblewski, J.: Rough Set Algorithms in Classification Problem. In: Rough Set Methods and Applications, pp. 49–88. Physica-Verlag, Heidelberg (2000)
5. Bazan, J.G., Szczuka, M.S.: RSES and RSESlib - A Collection of Tools for Rough Set Computations. In: Rough Sets and Current Trends in Computing, pp. 106–113 (2000)
6. Huhtala, Y., Karkkainen, J., Porkka, P., Toivonen, H.: TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies. The Computer Journal 42(2), 100–111 (1999)
7. Jelonek, J., Krawiec, K., Stefanowski, J.: Comparative study of feature subset selection techniques for machine learning tasks. In: Proceedings of IIS, Malbork, Poland, pp. 68–77 (1998)
8. Kryszkiewicz, M.: Algorytmy Redukcji Wiedzy w Systemach Informacyjnych, Ph.D. Thesis, Warsaw University of Technology (1994)
9. Kryszkiewicz, M.: Comparative Study of Alternative Types of Knowledge Reduction in Inconsistent Systems. Intl. Journal of Intelligent Systems 16(1), 105–120 (2001)
10. Kryszkiewicz, M.: Certain, Generalized Decision, and Membership Distribution Reducts versus Functional Dependencies in Incomplete Systems. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 162–174. Springer, Heidelberg (2007)
11. Kryszkiewicz, M., Cichon, K.: Towards Scalable Algorithms for Discovering Rough Set Reducts. In: Peters, J.F., Skowron, A., Grzymała-Busse, J.W., Kostek, B.z., Świniarski, R.W., Szczuka, M.S. (eds.) Transactions on Rough Sets I. LNCS, vol. 3100, pp. 120–143. Springer, Heidelberg (2004)
12. Kryszkiewicz, M., Lasek, P.: Fast Discovery of Minimal Sets of Attributes Functionally Determining a Decision Attribute. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS, vol. 4585, pp. 320–331. Springer, Heidelberg (2007)
13. Lin, T.Y.: Rough Set Theory in Very Large Databases. In: Proceedings of CESA IMACS, Lille, France, vol. 2, pp. 936–941 (1996)
14. Lopes, S., Petit, J.-M., Lakhal, L.: Efficient Discovery of Functional Dependencies and Armstrong Relations. In: Zaniolo, C., Grust, T., Scholl, M.H., Lockemann, P.C. (eds.) EDBT 2000. LNCS, vol. 1777, pp. 350–364. Springer, Heidelberg (2000)
15. Nguyen, S.H., Skowron, A., Synak, P., Wroblewski, J.: Knowledge Discovery in Databases: Rough Set Approach. In: Proceedings of IFSA, Prague, vol. 2, pp. 204–209 (1997)

16. Ohrn, A.: Discernibility and Rough Sets in Medicine: Tools and Applications, PhD thesis, Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway. NTNU report 1999:133, IDI report 1999:14
17. Pancerz, K., Suraj, Z.: Rough sets for discovering concurrent system models from data tables. In: Rough Computing: Theories, Technologies and Applications, pp. 239–268. Idea Group, Inc. (2007)
18. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning about Data, vol. 9. Kluwer Academic Publishers, Dordrecht (1991)
19. Pawlak, Z.: Concurrent versus sequential the rough sets perspective. Bulletin of the EATCS 48, 178–190 (1992)
20. Shan, N., Ziarko, W., Hamilton, H.J., Cercone, N.: Discovering Classification Knowledge in Databases Using Rough Sets. In: Proceedings of KDD, pp. 271–274 (1996)
21. Skowron, A.: Boolean Reasoning for Decision Rules Generation. In: Komorowski, J., Raś, Z.W. (eds.) ISMIS 1993. LNCS, vol. 689, pp. 295–305. Springer, Heidelberg (1993)
22. Skowron, A., Suraj, Z.: Discovery of Concurrent Data Models from Experimental Tables: A Rough Set Approach. In: Proceedings of KDD, Montreal, Canada, pp. 288–293 (1995)
23. Slezak, D.: Approximate Reducts in Decision Tables. In: Proceedings of IPMU, Granada, Spain, vol. 3, pp. 1159–1164 (1996)
24. Slezak, D.: Searching for Frequential Reducts in Decision Tables with Uncertain Objects. In: Polkowski, L., Skowron, A. (eds.) RSCTC 1998. LNCS, vol. 1424, pp. 52–59. Springer, Heidelberg (1998)
25. Slezak, D.: Association Reducts: Complexity and Heuristics. In: Greco, S., Hata, Y., Hirano, S., Inuiguchi, M., Miyamoto, S., Nguyen, H.S., Słowiński, R. (eds.) RSCTC 2006. LNCS, vol. 4259, pp. 157–164. Springer, Heidelberg (2006)
26. Slowinski, R. (ed.): Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory, vol. 11. Kluwer Academic Publishers, Dordrecht (1992)
27. Stepaniuk, J.: Approximation Spaces, Reducts and Representatives. In: Rough Sets in Data Mining and Knowledge Discovery. Springer, Berlin (1998)
28. Suraj, Z.: Rough Set Methods for the Synthesis and Analysis of Concurrent Processes. In: Rough set methods and applications. New developments in knowledge discovery in information systems (Studies in fuzziness and soft computing), pp. 379–488. Physica-Verlag, Heidelberg (2000)
29. Wroblewski, J.: Finding Minimal Reducts Using Genetic Algorithms. In: Proceedings of JCIS, Wrightsville Beach, NC, September/October 1995, pp. 186–189 (1995)

Information Granulation: A Medical Case Study

Urszula Kużelewska and Jarosław Stepaniuk

Department of Computer Science
Białystok University of Technology
Wiejska 45a, 15-351 Białystok, Poland
{uk,jstepan}@wi.pb.edu.pl

Abstract. Granular computing is a multidisciplinary theory rapidly developed in recent years. It provides a conceptual framework for many research fields, among others data mining. Data mining techniques and algorithms focus on knowledge discovery from data. When data labels are unknown one can use methods of exploratory data analysis called clustering algorithms. Clustering algorithms are also useful to find hidden dependencies and patterns in data. In this article granular computing and clustering were implemented in information granulation system SOSIG and applied to exploration of real medical data set. Data granulation in the system can be performed on different levels of resolution. Thereby the granules composed of clusters reflect relationship between objects on distinct levels of details. The clustering in SOSIG is generated automatically - there is no requirement to give a number of groups for division. It eliminates problems present in popular clustering algorithms like selection of correct number of clusters and evaluation of created partitioning. The difficulties are encountered in most partitioning as well as hierarchical methods reducing their practical application.

Additionally, this article contains solution generated by SOSIG in comparison with clustering results of algorithms: k-means, hierarchical, EM and DBSCAN. There are used quality indices such as Dunn's, DB, CDbw and SI.

Keywords: knowledge discovery and data mining, information granulation, clustering.

1 Introduction

Granular computing (GC) is based on processing of complex information entities called granules. Generally speaking, granules are collection of entities, that are arranged together due to their similarity, functional adjacency or indistinguishability [23]. Granulation information is not a specific set of methods or principles, it is an approach of analyzing data and its distinguishable aspect is multi-perspective point of view. Multi-perspective stands for diverse levels of resolution depending on saliency of features or grade of details of studied problem. At different data granularities, different properties and relationships appear. This is used in designing more effective algorithms applied in many disciplines and methods e. g. diakoptics, divide and conquer, structured programming, interval analysis, quantization, rough set theory, chunking, cluster analysis, machine learning, data mining [1]. To give more universal definition, granular computing may be considered as a label of a new field of multi-disciplinary study, dealing with theories, methodologies, techniques and tools that make use of granules in the process of problem solving [21].

Clustering can be viewed as a realization of GrC principles [12]. Grouping techniques determine natural clusters, that is groups of feature vectors or objects more similar to one another than to the objects from other clusters [7]. Criterion of similarity is dependent on clustering algorithm and data type. The most common similarity measure is distance between the points, for example, Euclidean for continuous attributes. From the machine learning standpoint, clustering is unsupervised learning - there are not available labels of data and often distribution of data is also unknown. Partitioning algorithms have had wide applications in pattern recognition, image processing, statistical data analysis and knowledge discovery.

According to a new framework of granular computing in data mining [17], [22] each component of a cluster is treated as a knowledge granule, whereas the cluster represents structural knowledge. Knowledge granule corresponds to a set of objects, in particular to the object itself, from the training data. Additionally, granules are expected to help interpret solved problem for humans. According to this, a set of semantically-described granules is a desirable result generated by any information granulation approach.

The paper presents granulation by clustering of the set concerning data of children suffering from diabetes disease. The granules are formed as a result of clustering performed on appropriate resolution level. To make them well-interpretable there is a step of semantic granules creation as a combination of the partitioning and discretization of the attributes according to medical norms for data attributes. There are generated and compared clustering results obtained from algorithms: k-means, hierarchical, EM [7], DBSCAN [4] and SOSIG. For measure of granules quality were applied Dunn's, DB [5], CDbw [6] and silhouette indices (SI) [8].

The paper is organized as follows. In Section 2 we recall SOSIG system [10] which is a successor of SArIS algorithm [20]. In Section 3 measures of granule quality are discussed. In Section 4 granulation of real life data (diabetes dataset) in form of semantically described granules is presented.

2 Self-Organizing System for Information Granulation

The SOSIG (**S**elf-**O**rganizing **S**ystem for **I**nformation **G**ranulation) algorithm is a data granulation system designed for detecting relationships and dependencies present in data. Granulation process can be performed on different levels of resolution depending on required details of solution. The resolution in SOSIG is regulated by rg parameter, however it strongly depends on the structure in

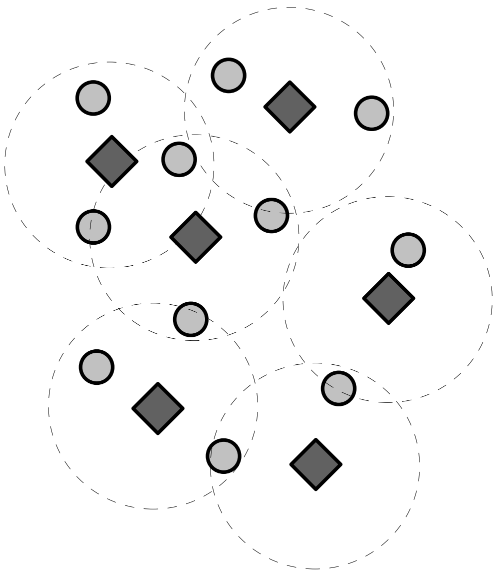


Fig. 1. Idea of representative objects in algorithm SOSIG (input data are marked by circles, network objects by rhombuses)

data. To be more precise, if the relationships between objects in data set form hierarchical structure, its levels can be identified by regulation of rg value, if the data are "flat", it is possible to create one-level granulation only.

Main part of SOSIG is based on SArIS algorithm [20], however there have been made major changes and adaptations (like manipulating different types of attributes, adjusting of solution, clustering new data) described in the following text.

SOSIG creates a network structure of connected points forming clusters. Organization of the system, including the points as well as the connections, is constructed on the basis of relationships between input data, without any external supervision. The structure points are representatives of input data, that is an individual object from the structure stands for one or more object from input set. The idea of representative objects is illustrated in Figure 1. In effect of this the number of representatives is much less than clustered data without lost of information. Steps of the algorithm are described in Algorithm 1. Additionally, initial part is separated into Algorithm 2, elimination not-useful objects into Algorithm 3 and adjusting of the solution into Algorithm 5. Labeling newly-added objects is presented in Algorithm 6.

Let us assume input data defined as an information system $IS = (U, A)$ [11], where $U = \{x_1, \ldots, x_n\}$ is a set of objects and $A = \{a_1, \ldots, a_k\}$ is a set of attributes.

Similarity between pair of objects is based on functions in different form for different types of attributes. General form of the dissimilarity is expressed by Equation 1, whereas practically it is calculated from Equation 2.

$$\delta(x,y) = fusion(\delta_{a_1}(a_1(x), a_1(y)), \ldots, \delta_{a_k}(a_k(x), a_k(y))). \tag{1}$$

$$\delta(x,y) = \frac{\delta_{real}(x,y) \cdot k_{real} + \delta_{nom}(x,y) \cdot k_{nom}}{k}. \tag{2}$$

According to the type of attribute, particular components of Equation 2 are calculated from Equations 3 and 4 for real and nominal attributes respectively.

$$\delta_{real}(x,y) = \sqrt{\sum_{i=1}^{k} (a_i(x) - a_i(y))^2}, \tag{3}$$

$$\delta_{nom}(x,y) = \frac{card(\{a \in A : a(x) \neq a(y)\})}{k_{nom}}. \tag{4}$$

Result generated by SOSIG is also described by an information system $IS' = (Y, A \cup \{a_{gr}\})$, where the last attribute $a_{gr} : Y \rightarrow \{1, \ldots, nc\}$ denotes label of generated granule and $card(Y) \leq card(U)$ and $\forall x \in U \exists y \in Y (\delta(x,y) < NR)$.

The parameter NR existing in above definition defines neighborhood region of objects from IS'. It directly influences level of granulation of the input set. Initial value of NR is proportional to maximal of nearest neighbor distances in the input set (see Equation 5).

$$NR_{init} = \max(\{\min(\{\delta(x_i, x_j) : x_j \in U \ \& \ \ x_j \neq x_i\}) : x_i \in U\}). \tag{5}$$

The following values of NR are calculated from current state of the network (see Equation 6).

$$NR = rg \cdot \frac{\sum_{y_i \in Y} \min(\{\delta(y_i, y_j) : y_j \in Y \ \& \ \ y_j \neq y_i\})}{card(Y)}, \tag{6}$$

where $rg \in [0,1]$ is a resolution of granulation parameter. The most often value of granulation resolution parameter is 0.5 (estimated theoretically and confirmed in experiments) assigned to the most separated clusters. Increasing the value of rg it is possible to identify granules in higher resolution. The NR directly affects cluster formation as connections in network are determined if the objects are in their neighborhoods.

After initial phase, like normalization of data, calculation initial value of NR (see Algorithm 2), iterated steps of the algorithm follow. First, the system objects are assessed. The measure of their usefulness is a similarity level s_l expressed by Equation 7.

$$s_l(y) = NR - \min(\{\delta(y,x) : x \in U\}). \tag{7}$$

The similarity level determines proximity degree of every representative to the training data distinguishing network objects in close location to input points.

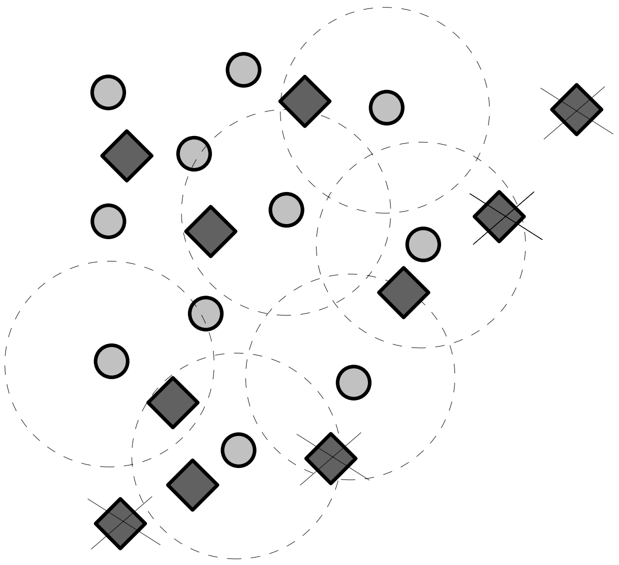


Fig. 2. Removing from the network not useful objects in algorithm SOSIG (input data are marked by circles, network objects by rhombuses)

Then not useful objects are removed (the procedure is presented in Algorithm 3). It affects extremely dissimilar objects as redundant ones. As redundant are determined points having the same input object in their neighborhood. The best of them stay in the network and also not redundant ones for other input data. This process controls the size of the network prevents forming excessively dense clusters. It results in compression phenomenon. The removing step is illustrated in Figure 2.

The remaining objects are re-connected and labeled. Components of the same granule have equal label, whereas the granule is determined by edges between the objects in the structure. Then there is calculated a new value of NR parameter (see Equation 6). When stopping criterion is met, the algorithm is stopped after connections reconstruction. Otherwise following steps are carried out. As a stopping criterion there is considered a stable state of the network, that is the state of small fluctuations of network size and value of NR.

The last step is a procedure of adjusting generated solution to the input data (see Algorithm 5). It concerns all network objects by replicating and modification of their attribute values. It allows searching better solution nearby examined network object.

There is also a step introducing to the system object from input data not recognized yet, described in Algorithm 4. This operation avoids leaving not represented area in the training set.

Further classification of new as well as training points can be performed using so-created structure (see Algorithm 6). To assign a label to considered object it is necessary to determine neighborhood objects from network structure. The

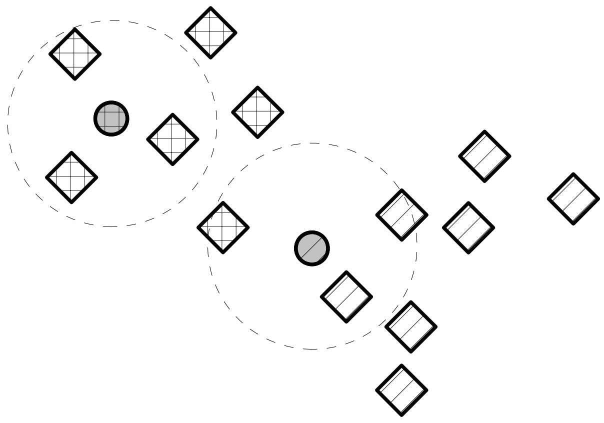


Fig. 3. Clustering newly-added objects in algorithm SOSIG (new input data are marked by circles, network objects by rhombuses, clusters are distinguished by different filling patterns)

neighborhood of the point is defined by final value of the NR (the last calculated value) of the SOSIG. The predominant value of the labels is given to the examined object. The procedure is presented in Figure 3.

The algorithm SOSIG is described in details in [10].

3 Quality of Information Granules

Together with specification of elementary granules it is necessary to define measures of granule quality [14]. In this article, there is considered clustering process as a realization of granular computing. The aim of clustering techniques is detection of granules, that are possibly the most compact and separable. To evaluate compactness and separation of discovered clusters there were proposed statistics so-called validity indices. Validity indices are designed to estimation of quality of obtained partitioning. Assessment the most optimal result needs calculation of validity indices for different values of algorithm's parameter, what usually is a number of clusters. The most commonly used are Dunn and Dunn-like statistics and Davies-Bouldin (DB) index [5]. Their advantage is exhibition no trends with respect to the number of clusters, therefore the minimum (DB) or maximum (Dunn) value indicate the most optimal partition.

Another index proposed by [8] to evaluate clustering result is Silhouette Index (SI). The Silhouette Index for the data set is the mean of this measure for all points in the data set and ranges from 1 to 1, where 1 indicates a good classification, -1 indicates a bad classification and 0 indicates that the classification could go either way. Since the Silhouette Index compares classification between clusters, it cannot be computed for single cluster data sets.

The mentioned above indices, although commonly used, suffer from a number of limitations. First of all, they are predisposed to cope with spherical

Algorithm 1. Construction of information system with a set of representative objects

Data:
- $IS = (U, A)$ - an information system, where $U = \{x_1, \ldots, x_n\}$ is a set of objects and $A = \{a_1, \ldots, a_k\}$ is a set of attributes,
- $\{\delta_a : a \in A\}$ - a set of distance function of the form $\delta_a : V_a \times V_a \to [0, \infty)$, where V_a is a set of values for attribute $a \in A$ and a global distance function $\delta : U \times U \to [0, \infty)$ defined by $\delta(x, y) = fusion(\delta_{a_1}(a_1(x), a_1(y)), \ldots, \delta_{a_k}(a_k(x), a_k(y)))$
- $size_{net} \in \{0, 1, \ldots, card(U)\}$ - initial size of network, $rg \in [0, 1]$ - resolution of granulation,

Result: $IS' = (Y, A \cup \{a_{gr}\})$ - an information system, where the last attribute $a_{gr} : Y \to \{1, \ldots, nc\}$ denotes label of generated granule and $card(Y) \leq card(U)$ and $\forall x \in U \exists y \in Y \delta(x, y) < NR$

```
begin
    $[NR_{init}, Y] \longleftarrow$ initialize($U, A, size_{net}$);
    for $y_i, y_j \in Y, i \neq j$ do /*form clusters*/
        if $\delta(y_i, y_j) < NR_{init}$ then connect($y_i, y_j$);
    $NR \longleftarrow NR_{init}$;
    while $\neg$stopIterations($Y$) do
        for $y \in Y$ do
            $\Delta(y) = (\delta(y, x))_{x \in U}$; /*calculate distances from input data*/;
            $s_l(y) = NR - \min \Delta(y)$;/*similarity level of the object*/;
        delete($U, A, Y$); /*remove redundant network objects*/;
        for $y_i, y_j \in Y, i \neq j$ do /* reconnect objects*/
            if $\delta(y_i, y_j) < NR$ then connect($y_i, y_j$);
            $a_{gr}(y_i) \longleftarrow 0$; $a_{gr}(y_j) \longleftarrow 0$;
        $grLabel \longleftarrow 1$;
        for $y_i \in Y$ do /*label objects*/
            if $a_{gr} = 0$ then $a_{gr}(y_i) \longleftarrow grLabel$;
            for $y_j \in Y, j \neq i$ do
                if connected($y_i, y_j$) then $a_{gr}(y_j) \longleftarrow grLabel$;
            $grLabel \longleftarrow grLabel + 1$;
        for $y_i \in Y$ do
            /*calculate the nearest neighbor network objects*/;
            $\delta_{NN}(y_i) = \min(\{\delta(y_i, y_j) : y_j \in Y \text{ \& } j \neq i\})$;
        $NR \longleftarrow rg \cdot \frac{\sum_{y \in Y} \delta_{NN}(y)}{card(Y)}$;/*new value of NR*/;
        if $\neg$stopIterations($Y$)/*test the stopping condition */ then
            joinNotRepresented($U, Y, NR, \Delta$);
            adjust($Y, U, A, NR$);
end
```

distribution of clusters and are likely to be data dependent. There are also difficulties when clusters are of not equal size and densities [5].

Recently proposed index CDbw (composing density between and with clusters) [6], on the contrary to others measures, uses multi-representatives points

Algorithm 2. Initial steps of SOSIG algorithm

```
Data: Set of input objects: $U = \{x_1, \ldots, x_n\}$, $size_{net} \in \{0, 1, \ldots, card(U)\}$ -
      initial size of network
Result: Set of initial network objects $Y = \{y_1, \ldots, y_{size_{net}}\}$, where $Y \subset U$,
        $NR_{init}$ - initial value of neighborhood radius threshold
begin
    $\delta_{maxNN} = 0$;
    for $x_i \in U$ do
        /*calculate the nearest neighbor distances of the data*/ ;
        $\delta_{NN}(x_i) = \min(\{\delta(x_i, x_j) : x_j \in U \ \& \ \ x_j \neq x_i\})$ ;
        /*find the greatest value of the nearest neighbor distances*/ ;
        if $\delta_{NN}(x_i) > \delta_{maxNN}$ then $\delta_{maxNN} = \delta_{NN}(x_i)$;
    $NR_{init} \longleftarrow \delta_{maxNN}$;
    /*select the representatives objects*/ ;
    for $netObj \leftarrow 1$ to $size_{net}$ do
        $i = rand(1, \ldots, card(U))$;
        $y_{netObj} = x_i$;
end
```

Algorithm 3. Detailed steps of *delete* function from SOSIG

```
Data: Set of input objects $U$, network set $Y$, set of attributes $A = \{a_1, \ldots, a_k\}$,
      $NR$ - threshold of neighborhood radius
Result: $Y \setminus C$, where $C$ is a set of redundant network objects
begin
    $C \longleftarrow \emptyset$;/*initially the set is empty*/;
    for $x \in U$ do
        for $y \in Y$ do
            if $\delta(y, x) < NR$ then /*add to the set objects representing the input
            element $x$*/;
            $C \longleftarrow C \cup \{y\}$;
        $\delta_{NN}(x, y_{NN}) = \min(\{\delta(x, y) : y \in Y\})$ ;
        $C \longleftarrow C \setminus \{y_{NN}\}$;/*remove from the set the best object representing $x$ */;
    $Y \longleftarrow Y \setminus C$;/*remove from the network objects from the set $C$*/;
end
```

to describe clusters, what enables evaluation groups of arbitrary shape. The CDbw index is based on cluster compactness and separation. It is product of two components $Sep(nc)$ and $Intra\_dens(nc)$ (see Equation 8).

$$CDbw(nc) = Sep(nc) \cdot Intra\_dens(nc), nc > 1, \tag{8}$$

where nc denotes a number of clusters in examined partitioning. $Intra\_dens(nc)$ depends on density of granules (described in the following text) and $Sep(nc)$ is a measure of separability of clusters defined as follows:

$$Sep(nc) = \frac{\sum_{i=1}^{nc} \sum_{j=1, i \neq j}^{nc} \min d(clos\_rep_i, clos\_rep_j)}{1 + Inter\_dens(nc)}, nc > 1. \tag{9}$$

Algorithm 4. Detailed steps of *joinNotRepresented* function from SOSIG

```
Data: Set of input objects: $U = \{x_1, \dots, x_n\}$, $Y = \{y_1, \dots, y_n\}$, $\Delta$ - matrix of
      distances between input and network objects, $NR$ - threshold of
      neighborhood radius
Result: $Y \cup \{x\}$ (with condition $\neg\exists y \in Y \delta(y, x) < NR$)
begin
    for $x \in U$ do
        /*find an arbitrary object from the training set not represented yet by
        any network element*/ ;
        $add \longleftarrow 1$ ;
        for $y \in Y$ do
            if $\delta(y, x) < NR$ then $add \longleftarrow 0$ ;
            break;
        if $add = 1$ then $Y \longleftarrow Y \cup \{x\}$;
        break;
end
```

Algorithm 5. Detailed steps of *adjust* function from SOSIG

```
Data: Set of input objects $U$, network set $Y$, set of attributes $A = \{a_1, \dots, a_k\}$,
      $NR$ - threshold of neighborhood radius
Result: $Y \cup Z$, where $Z$ is a set of adjusted network objects
begin
    $Z \longleftarrow \emptyset$;
    for $y \in Y$ do
        for $candidate \longleftarrow 1$ to $noCandidates$ do
            $z_{candidate} \longleftarrow y$;
            for $a \in A$ do
                $sign \longleftarrow rand(\{-1, 1\})$; $delta \longleftarrow sign$;
                if $a(z_{candidate})$ is binary then
                    /*value modification for binary attribute*/;
                    $randVal \longleftarrow rand(\{0, 1\})$; $delta \longleftarrow delta \cdot randVal$;
                    $a(z_{candidate}) \longleftarrow delta$;
                else
                    /*value modification for continuous attribute*/;
                    $randVal \longleftarrow rand([0, 1])$;
                    /*scale of change depends on the value od $s_l$ */;
                    $delta \longleftarrow delta \cdot randVal \cdot (1.5 - s_l(y))$;
                    $a(z_{candidate}) \longleftarrow a(z_{candidate}) + delta$;
            for $x \in U$ do
                if $\delta(z_{candidate}, x) < NR$ then
                    /*only useful clones are joined to the network */;
                    $Z \longleftarrow Z \cup \{z_{candidate}\}$;
                    break;
end
```

Algorithm 6. Clustering of the input information system in SOSIG algorithm

```
Data:
  – $IS = (U, A)$ - an information system, where $U = \{x_1, \dots, x_n\}$ is a set of
    objects and $A = \{a_1, \dots, a_k\}$ is a set of attributes,
  – $IS' = (Y, A \cup \{a_{gr}\})$ where last attribute $a_{gr} : Y \rightarrow \{1, \dots, ng\}$
    stands for label of generated granule and $card(Y) \leq card(U)$
  – $NR$ - threshold of neighborhood radius;
Result: Clustered information system $IS_{gr} = (U, A \cup \{a_{gr}\})$ into $ng$ clusters
        (granules), where last attribute $a_{gr} : U \rightarrow \{1, \dots, nc\}$ stands for label of
        generated granule
begin
    for $x \in U$ do
        for $granule \leftarrow 1$ to $ng$ do
            $grLabels[granule] \leftarrow 0$;
        /*calculate distance between $x$ and the network objects*/;
        for $y \in Y$ do
            if $\delta(x, y) < NR$ then  $label \leftarrow a_{gr}(y)$;
            $grLabels[label] \leftarrow grLabels[label] + 1$;
        /*predominant label is selected */;
        $a_{gr}(x) \leftarrow \max(\{grLabels\})$ ;
end
```

Separability is proportional to the sum of distances between the closest representative points (*close_rep*) from pair-wise clusters and inversely proportional to measure of density between clusters ($Inter\_dens(nc)$). Representative points are objects from a training set selected by Farthest First (FF) algorithm for every cluster. The FF algorithm works as follows: initially the cluster centroid is determined. Then there is selected the first representative point from the training data located the farthest from the centroid. In the following steps are selected the farthest objects from the previously determined representatives. The steps are repeated until the required number r of representatives is reached. Density between clusters is expressed by Equation 10. It is desirable to generate partitioning of the lowest inter-cluster density.

$$Inter\_dens(nc) = \sum_{i=1}^{nc} \sum_{j=1, j \neq i}^{nc} \left(\frac{d(clos\_rep_i, clos\_rep_j)}{stdev\,(C_i) + stdev\,(C_j)} \cdot density(u_{ij}) \right), nc > 1. \tag{10}$$

The component $stdev\,(C)$ is the standard deviation of cluster $C \subseteq U$ and $density(u_{ij})$ density of input objects around the point u_{ij} defined by Equation 11. The point u_{ij} is the middle point of the line segment defined by the closest clusters' representatives $close\_rep_i$ and $close\_rep_j$ and n_i, n_j are the number of objects in clusters C_i and C_j.

$$density(u_{ij}) = \frac{\sum_{x \in C_i \cup C_j} f(x, u_{ij})}{|C_i| + |C_j|}. \tag{11}$$

The $density(u_{ij})$ represents the percentage of points in the cluster i and the cluster j that belong to the neighborhood of u_{ij}. This neighborhood is defined to be a hyper-sphere with center u_{ij} and radius the average standard deviation of the clusters between which the density is estimated. The function $f(x, u_{ij})$ is defined as:

$$f(x, u_{ij}) = \begin{cases} 0 \text{ if } d(x, u_{ij}) > stdev\,(C_i) + stdev\,(C_j))/2 \\ 1 \text{ otherwise.} \end{cases} \tag{12}$$

The second component of Equation 8 determines the average density within clusters and is defined as the percentage of points that belong to the neighborhood of representative points of the considered clusters. The goal is the density within clusters to be significant high. *Intra_dens*(nc) is given by the following equation:

$$Intra\_dens(nc) = \frac{1}{nc} \sum_{i=1}^{nc} \frac{1}{r} \sum_{v_{ij} \in C_i} \frac{density(v_{ij})}{stdev\,(C_i)}, nc > 1, \tag{13}$$

where

$$density(v_{ij}) = \sum_{x \in C_i} g(x, v_{ij}). \tag{14}$$

The function $g(x, v_{ij})$ is described by Equation 15.

$$g(x, v_{ij}) = \begin{cases} 0 \text{ if } d(x, v_{ij}) > stdev\,(C_i) \\ 1 \text{ otherwise.} \end{cases} \tag{15}$$

To determine a good clustering scheme it is required to find a maximum value of CDbw.

4 Information Granulation in Medical Data

The aim of the carried out experiments is to present exploration of real life data. The information granules obtained in clustering process are subjected to the analysis. To create the granules there were selected algorithms k-means, hierarchical minimum-variance (hmv), hierarchical complete-link (hcl), EM, DBSCAN and SOSIG. Granulation schemes were assessed and compared by quality indices. Then medical norms of numerical attributes are employed to construct final semantic form of selected the best granules.

4.1 Description of the Dataset

The dataset used in experiments contains real life medical data described and examined in [15] and [16] (see also [9]). The database is shown at the end of the paper [16]. Results of granulation of not normalized the data has been presented in the article [18].

Table 1. Description of attributes of diabetes data. The last column contains real nominal attribute values or discretized values according to medical norms for numerical attributes.

Attr	Description of attribute	Type of attribute	Values of attributes
a_1	Sex	Nominal	m, f
a_2	Age of diagnosis (years)	Numerical	$< 7, 7-12, 13-15, \geq 16$
a_3	Disease duration (years)	Numerical	$< 6, 6-10, \geq 11$
a_4	Diabetes in family	Nominal	yes, no
a_5	Insulin therapy type	Nominal	KIT, KIT_IIT
a_6	Resp. system infections	Nominal	yes, no
a_7	Remission	Nominal	yes, no
a_8	HbA1c	Numerical	$< 8, [8, 10), \geq 10$
a_9	Hypertension	Nominal	yes, no
a_{10}	Body mass (percentiles)	Nominal	$< 3, 3-97, > 97$
a_{11}	Hypercholesterolemia	Nominal	yes, no
a_{12}	Hypertriglycerodemia	Nominal	yes, no
a_{13}	Microalbuminuria	Nominal	yes, no

There are 107 objects containing information about children suffering from diabetes. Detailed description of attributes is placed in Table 1 (see also [17]). One of the attributes is in fact a decision attribute, and in the experiments we took into consideration two variants of the data: with and without the attribute. The attributes include sex, the age at which the disease was diagnosed and other diabetological findings. There are also criteria of the metabolic balance, hypercholesterolemia and hypertriglyceridemia. Most of them have nominal values and there are also continuous real attributes.

4.2 Results of the Experiments

To eliminate influence of attributes having the highest variance in their values normalization process on all of them has been performed. The attributes have been normalized to interval $[0, 1]$. The clustering was executed twice: with additional attribute - the class attribute (microalbuminuria) present and without it. The algorithms k-means, hmv and EM have been run for $nc = 2, 3, ..., 10$, the method SOSIG for various values of granulation resolution rg and DBSCAN for various values of parameter $epsilon$ with $minPts = 3$ (the $epsilon$ parameter stands for radius of density neighborhood and minPts denotes minimal number of points in single cluster). Every partitioning is assessed by indices Dunn's, DB, SI and CDbw. The results containing less than 90% objects from input set clustered were not taken into consideration.

Left side of Table 2 presents evaluation of granulation of the 13 dimensional diabetes data, whereas right side evaluation of granulation of the same set excluding the class attribute microalbuminuria. There is high discrepancy in optimal partitioning indicated by the values in Table 2. The most often distinguished is clustering containing 8, 9 and 10 granules, however there are also indicated results containing 2, 3 and 4 groups.

In DBSCAN result for $epsilon < 1.1$ regardless of minPts value only 20-30 objects were clustered, whereas for $epsilon > 1.1$ all input objects were assigned

Table 2. Evaluation of clusterings of the diabetes data including (left side) and excluding (right side) the class column created by parametrical algorithms: k-means, hmv and EM

Method	Index	Best value	nc	Method	Index	Best value	nc
	Dunn	0.43	6-9		Dunn	0.48	10
k-means	DB	1.51	10	k-means	DB	1.35	9
	SI	0.18	10		SI	0.19	9,10
	CDbw	1.82	3		CDbw	2.18	2
	Dunn	0.43	9,10		Dunn	0.48	9,10
hmv	DB	1.53	10	hmv	DB	1.35	10
	SI	0.17	10		SI	0.18	8,10
	CDbw	1.72	2		CDbw	2.18	2
	Dunn	0.37	3,4		Dunn	0.14	9
EM	DB	1.79	10	EM	DB	1.38	4
	SI	0.15	2		SI	0.16	2
	CDbw	1.85	3		CDbw	1.16	2

to one cluster. The only partitioning is obtained for *epsilon* = 1.1 containing 2 groups of 99 and 8 elements. Calculated assessment indices were DB=2.24, Dunn=0.36, CDbw=1.90, SI=0.12. Since high difference in sizes of clusters and low quality indicated the result is not taken in further considerations.

Similar results were obtained in SOSIG granulation. There was also threshold value of rg parameter ($rg = 0.45$ for 13 and $rg = 0.50$ for 12 dimensional set), where over the value the partitioning was composed of all objects forming one cluster, whereas below the threshold value there were predominant clusters consisted of single objects in the result.

Considering the examples described before and ambiguous results of granulation 13 and 12 attributes of diabetes data one can assume there is absence of compact and separable clusters in the data. Presence of microalbuminuria attribute exerts an influence on obtained clustering, however separability of groups is low in both cases. This is concluded on the basis of highly diverse assessment values for parametrical clustering methods k-means, hierarchical and EM. More effortless and unambiguous interpretation is in case of SOSIG result. There is or not enough number of input objects in large clusters (what suggests the value of rg parameter is too high) or all the objects are forming one cluster. Formation one group implies the objects are equally similar to one another.

In further exploration of the set selection of attributes step has been executed. Importance of attributes can be measured by different methodologies. Three of them were presented in [15] and applied to the considered diabetes data set. The paper describes reducts application, a method based on significance of attributes and a method inspired by wrapper approach. To select attributes for further granulation there were prepared 11 data sets ($S_1 \ldots S_{11}$) containing the attributes from experiments from the paper [15]. There were taken into consideration 6 reducts ($S_1 \ldots S_6$) - the 2 best subsets indicated by the method based on significance of attributes (S_7, S_8) and 2 subsets evaluated as the best by the method inspired by wrapper approach (S_9, S_{10}). There was also examined a subset containing 3 continuous variables (S_{11}). Composition of the subsets is shown in Table 3.

Table 3. Composition of considered subsets $S_1 \ldots S_{11}$

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
S_1	+	+	+	-	+	+	-	+	+	+	+	+
S_2	+	+	+	+	+	-	+	+	-	-	+	+
S_3	+	+	+	+	+	-	+	+	-	+	-	+
S_4	+	+	+	+	+	-	+	+	+	-	-	+
S_5	+	+	+	+	+	+	-	+	-	+	+	+
S_6	+	+	+	+	+	+	-	+	+	+	-	+
S_7	+	+	+	-	+	-	-	+	-	-	-	+
S_8	+	+	+	-	+	-	-	+	-	-	-	-
S_9	+	+	+	+	+	-	+	+	-	+	+	+
S_{10}	+	+	+	+	+	-	+	+	-	-	-	-
S_{11}	-	+	+	-	-	-	-	+	-	-	-	-

Table 4. Evaluation of subsets of the diabetes data granulation created by SOSIG algorithm, number of granules (ng)

	S_2	S_3	S_4	S_7	S_8	S_{11}
ng	6	8	7	5	4	4
DB	0.11	0.35	0.37	0.17	0.20	0.19
Dunn	1.13	0.71	0.70	1.00	0.95	0.17
SI	0.70	0.57	0.61	0.65	0.63	0.25
CDbw	7.48	0.60	2.32	15.18	18.36	2.29

So-prepared sets were clustered by SOSIG and DBSCAN algorithms. Then values of the quality indices DB, Dunn's, SI and CDbw were calculated. Tables 4 and 5 contain the number of groups and quality results of the partitionings of SOSIG and DBSCAN respectively. There were taken into consideration groups of minimal size equal 3. It was not possible to generate results for subsets $S_1, S_5, S_6, S_9, S_{10}$ by SOSIG and for subsets S_5, S_6, S_{11} by DBSCAN method. The reason is existence groups not distinct enough.

Analyzing the results one can notice the algorithms generate similar partitioning. Number of detected groups and quality of created clustering are comparable. For further experiments there was selected the clustering of subset S_7 performed by SOSIG due to optimal values of the quality indices for both algorithms and relatively small number of created groups. The following results concern composition of granules and rules generated by RSES system.

The partitioning of S_7 subset generated by SOSIG is composed of 5 large granules containing 87 input objects. The size of the two largest granule is 27 and 22. The remaining granules are composed of 16, 14 and 6 objects. Composition of values in every granule is presented in Table 6.

To fulfil the postulate of GrC to design systems delivering easy interpretable knowledge, the granules were used in RSES system to create granules in form of semantic rules. There were additionally applied medical norms (from Table 1) to discretize continuous and integer variables. The generated sets of rules present Table 7.

The granules described by rule set from RSES system have compact and transparent form, thereby the contained knowledge is more effective to learn.

Table 5. Evaluation of subsets of the diabetes data granulation created by DBSCAN algorithm, number of granules (ng)

	S_1	S_2	S_3	S_4	S_7	S_8	S_9	S_{10}
ng	24	6	10	8	7	4	21	12
DB	1.18	1.06	0.42	1.18	0.20	0.25	1.10	0.29
Dunn	0.41	0.41	0.55	0.39	0.86	0.86	0.41	0.63
SI	0.46	0.47	0.51	0.49	0.60	0.59	0.43	0.58
CDbw	0.51	0.88	1.00	0.78	3.12	5.24	0.79	1.69

Table 6. Granules generated by SOSIG

Granule (size)	Value	a_1	a_2	a_3	a_5	a_8	a_{12}
	minimal	f	9	3	KIT_IIT	6.95	no
1(16)	maximal	f	17	7	KIT_IIT	11.61	no
	average	-	12.44	5.25	-	8.69	-
	minimal	f	2	3	KIT	5.0	no
2(22)	maximal	f	16	8	KIT	9.6	no
	average	-	9.27	5.86	-	7.48	-
	minimal	m	8	4	KIT_IIT	5.8	no
3(14)	maximal	m	18	7	KIT_IIT	9.69	no
	average	-	11.71	4.93	-	8.11	-
	minimal	m	2	2	KIT	6.21	no
4(27)	maximal	m	16	7	KIT	10.08	no
	average	-	10.07	5.29	-	7.88	-
	minimal	m	2	6	KIT	8.83	yes
5(6)	maximal	m	12	8	KIT	10.8	yes
	average	-	7.5	6.83	-	10.06	-

Table 7. Decision rules generated by RSES system on the basis of granules created by SOSIG and medical norms applied to diabetes data

No of the rule	**if** Condition **then** Granule number	Size
1	**if** $a_1 = m$ **and** $a_5 = KIT$ **and** $a_{12} = no$ **then** $Granule = 4$	27
2	**if** $a_1 = f$ **and** $a_5 = KIT$ **then** $Granule = 2$	22
3	**if** $a_1 = f$ **and** $a_5 = KIT\_IIT$ **then** $Granule = 1$	16
4	**if** $a_1 = m$ **and** $a_5 = KIT\_IIT$ **then** $Granule = 3$	14
5	**if** $a_1 = m$ **and** a_3 =average **and** $a_5 = KIT$ **and** $a_{12} = yes$ **then** $Granule = 5$	6

The semantic form of granules is well-interpretable by the user without any special training.

Granulation of the diabetes data can be also used to evaluate risk of complication occurrence in the particular groups. In the examined data set the complication is microalbuminuria, described in Table 1 as a_{13} attribute. This action effects in indicating groups of patients particulary subjected to diabetes complication. Distribution of the decision attribute among the groups in presented in Figure 4. There are 3 the most homogeneous granules, that is granules with number 1,4,5. The granule 1 contains 88% cases of positive microalbuminuria, the granule with number 5 concerns 66% positive cases, whereas the granule 4 is predominated by 66% cases of microalbuminuria absence.

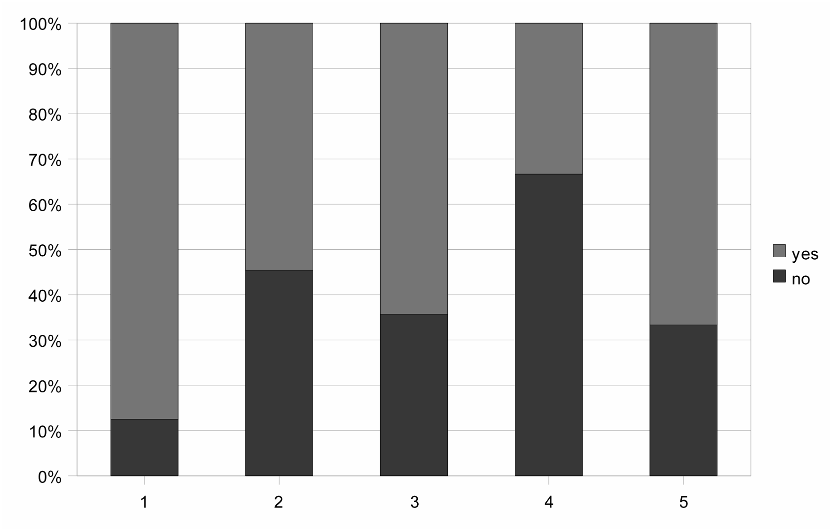


Fig. 4. Distribution of microalbuminuria attribute in particular granules

Among the granules the group number 1 has particularly high risk of the complication. It is composed of girls of age 9-17 undergone KIT_IIT therapy. Disease duration is between 3 and 7 years, whereas HbA1c level interval between [6.75,11.61]. The children belonging to this groups do not suffer from hipertriglycerodemia. The group number 5 consists boys of age between 2 and 12 has high risk of microalbuminuria. The patients, suffering from diabetes 6-8 years, were given a treatment by KIT therapy. An interval of HbA1c level in this group is relatively narrow - between 8.83 and 10.08. This is the only granule with hipertriglycerodemia occurrence. Boys are also included in the group number 4. This is the most numerous granule containing patients of age 2-16 suffering from the disease 2-7 years. The level of HbA1c is between [6.21,10.08]. The group has low risk of the complication. It is similar to granule 5, however there is hipertriglycerodemia absent. It may be concluded, that there is a connection between hipertriglycerodemia and microalbuminuria when patient is a boy suffering from diabetes (see the last rule from Table 7).

Analyzing the remaining granules one can notice a similarity between granules 2 and 1, however in the first of them the risk of complication is strongly reduced (from 88% to about 55%). The difference is also another therapy type - KIT. One can suggest, that the therapy KIT is more effective in group of girls suffering from diabetes. Group number 3 is composed of male patients of age 2-16, with disease duration between 4 and 7 years. The HbA1c level interval is between [5.8,9.69]. The children underwent KIT_IIT therapy, however there is quite high risk of microalbuminuria on contrary to similar group number 4. It may result from different type of treatment and there may also appear the suggestion of more effective KIT therapy in case of male patients.

5 Conclusions

The SOSIG algorithm is designed for knowledge discovery from data. The knowledge is described as information granules composed of similar to each other objects of data. The algorithm has been tested on a real multidimensional database. To balance an influence of all attributes on final granulation normalization process has been performed. When data set contains all 13 attributes distinct granules are not possible for identification. It is easy to conclude from SOSIG result, where all objects form one cluster - the contrary to result of parametrical algorithms where its assessment is very complex and ambiguous.

Further preparation of the data allowed identify granules present in data. Granules generated by SOSIG system are created unambiguous avoiding difficult phase of results selection. The granules are characterized by distinguishable values of the attributes. The final form of the results contains knowledge easy to interpret by medical doctors. The knowledge is presented in form of semantically described granules and set of rules extracted from RSES system applying additionally medical norms for the continuous and integer attributes.

There are homogenous granules with respect to decision attribute - microalbuminuria in the generated solution. One of them is characterized extremely high risk of the complication. It can be helpful in evaluation of possibility of further complication in new patients as well as selecting appropriate therapy type.

Acknowledgements

The research is supported by the grant N N516 0692 35 from Ministry of Science and Higher Education of the Republic of Poland. The research of Urszula Kuzelewska is supported by the Rector's of Technical University of Bialystok Grant No. W/WI/5/07.

References

1. Bargiela, A., Pedrycz, W.: Granular Computing: an Introduction. Kluwer Academic Publishers, Boston (2002)
2. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York (1981)
3. Cox, T.F., Cox, M.A.: Multidimensional Scaling, Monographs on Statistics and Applied Probability. Chapman-Hall, London (1994)
4. Ester, M., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of 2nd International Conference Knowledge Discovery and Data Mining, pp. 226–231. AAAI-Press, Portland (1996)
5. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. Journal of Intelligent Information Systems 17(2/3), 107–145 (2001)
6. Halkidi, M., Vazirgiannis, M.: Clustering validity assessment using multi representatives. In: Vlahavas, I.P., Spyropoulos, C.D. (eds.) SETN 2002. LNCS, vol. 2308. Springer, Heidelberg (2002)

7. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys 31(3), 264–323 (1999)
8. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, Chichester (1990)
9. Kryszkiewicz, M., Lasek, P.: Fast Discovery of Minimal Sets of Attributes Functionally Determining a Decision Attribute. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS, vol. 4585, pp. 320–331. Springer, Heidelberg (2007)
10. Kużelewska, U.: Data exploration by clustering algorithms with information granulation (in Polish), Ph.D (in preparation)
11. Pawlak, Z.: Rough Sets. Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht (1991)
12. Pedrycz, W.: Knowledge Based Clustering. From Data to Information Granules. John Wiley & Sons, Chichester (2005)
13. RSES (Rough Set Exloration System), `logic.mimuw.edu.pl/~rses/`
14. Skowron, A., Stepaniuk, J.: Modeling of High Quality Granules. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS, vol. 4585, pp. 301–310. Springer, Heidelberg (2007)
15. Stepaniuk, J.: Rough Set Data Mining of Diabetes Data. In: Raś, Z.W., Skowron, A. (eds.) ISMIS 1999. LNCS, vol. 1609, pp. 457–465. Springer, Heidelberg (1999)
16. Stepaniuk, J.: Knowledge discovery by application of rough set models. In: Polkowski, L., Tsumoto, S., Lin, T.Y. (eds.) Rough Set Methods and Applications. New Developments in Knowledge Discovery in Information Systems, pp. 137–233. Physica–Verlag, Heidelberg (2000)
17. Stepaniuk, J.: Rough–Granular Computing in Knowledge Discovery and Data Mining. Springer, Heidelberg (2008)
18. Stepaniuk, J., Kużelewska, U.: Granulation using clustering: A medical case study. In: Proceedings of International Conference on Concurrency, Specification and Programming, vol. 2, pp. 509–520 (2007)
19. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
20. Wierzchoń, S.T., Kużelewska, U.: Evaluation of clusters quality in artificial immune clustering system - SArIS. In: Biometrics, computer security systems and artificial intelligence applications, pp. 323–331. Springer, Heidelberg (2006)
21. Yao, Y.Y.: Information granulation and rough set approximation. International Journal of Intelligent Systems 16, 87–104 (2001)
22. Yao, Y.Y.: Granular computing for data mining. In: Dasarathy, B.V. (ed.) Proceedings of SPIE Conference on Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security, Kissimmee, Florida, USA, pp. 1–12 (2006)
23. Zadeh, L.A.: A new direction in AI: Toward a computational theory of perceptions. AI Magazine 22(1), 73–84 (2001)

Maximum Class Separability for Rough-Fuzzy C-Means Based Brain MR Image Segmentation

Pradipta Maji and Sankar K. Pal

Center for Soft Computing Research, Machine Intelligence Unit
Indian Statistical Institute, 203 B. T. Road, Kolkata, 700 108, India
{pmaji,sankar}@isical.ac.in

Abstract. Image segmentation is an indispensable process in the visualization of human tissues, particularly during clinical analysis of magnetic resonance (MR) images. In this paper, the rough-fuzzy *c*-means (RFCM) algorithm is presented for segmentation of brain MR images. The RFCM algorithm comprises a judicious integration of the rough sets, fuzzy sets, and *c*-means algorithm. While the concept of lower and upper approximations of rough sets deals with vagueness and incompleteness in class definition of brain MR images, the membership function of fuzzy sets enables efficient handling of overlapping classes. The crisp lower bound and fuzzy boundary of a class, introduced in the RFCM algorithm, enable efficient segmentation of brain MR images. One of the major issues of the RFCM based brain MR image segmentation is how to select initial prototypes of different classes or categories. The concept of discriminant analysis, based on the maximization of class separability, is used to circumvent the initialization and local minima problems of the RFCM. Some quantitative indices are introduced to extract local features of brain MR images for accurate segmentation. The effectiveness of the RFCM algorithm, along with a comparison with other related algorithms, is demonstrated on a set of brain MR images.

Keywords: Rough sets, fuzzy sets, medical imaging, segmentation, *c*-means algorithm.

1 Introduction

Segmentation is a process of partitioning an image space into some non-overlapping meaningful homogeneous regions. The success of an image analysis system depends on the quality of segmentation [1]. If the domain of the image is given by Ω, then the segmentation problem is to determine the sets $S_k \subset \Omega$, whose union is the entire domain Ω. The sets that make up a segmentation must satisfy

$$\Omega = \bigcup_{k=1}^{K} S_k, \text{ where } S_k \cap S_j = \emptyset \text{ for } k \neq j,$$

and each S_k is connected. Thus, a segmentation method is supposed to find those sets that correspond to distinct anatomical structures or regions of interest in the image. In the analysis of medical images for computer-aided diagnosis and therapy, segmentation is often required as a preliminary stage. However, medical image segmentation is a complex and challenging task due to intrinsic nature of the images. The brain has a particularly complicated structure and its precise segmentation is very important for detecting tumors, edema, and necrotic tissues, in order to prescribe appropriate therapy [2].

In medical imaging technology, a number of complementary diagnostic tools such as x-ray computer tomography (CT), magnetic resonance imaging (MRI), and position emission tomography (PET) are available. Magnetic resonance imaging (MRI) is an important diagnostic imaging technique for the early detection of abnormal changes in tissues and organs. Its unique advantage over other modalities is that it can provide multispectral images of tissues with a variety of contrasts based on the three MR parameters ρ, T1, and T2. Therefore, majority of research in medical image segmentation concerns MR images [2].

Conventionally, the brain MR images are interpreted visually and qualitatively by radiologists. Advanced research requires quantitative information, such as the size of the brain ventricles after a traumatic brain injury or the relative volume of ventricles to brain. Fully automatic methods sometimes fail, producing incorrect results and requiring the intervention of a human operator. This is often true due to restrictions imposed by image acquisition, pathology and biological variation. So, it is important to have a faithful method to measure various structures in the brain. One of such methods is the segmentation of images to isolate objects and regions of interest.

Many image processing techniques have been proposed for MR image segmentation, most notably thresholding [3, 4], region-growing [5], edge detection [6], pixel classification [7, 8] and clustering [9, 10, 11]. Some algorithms using the neural network approach have also been investigated in the MR image segmentation problems [12, 13]. One of the main problems in medical image segmentation is uncertainty. Some of the sources of this uncertainty include imprecision in computations and vagueness in class definitions. In this background, the possibility concept introduced by the fuzzy set theory [14] and rough set theory [15] have gained popularity in modeling and propagating uncertainty. Both fuzzy set and rough set provide a mathematical framework to capture uncertainties associated with human cognition process [16, 17, 18]. The segmentation of MR images using fuzzy c-means has been reported in [9, 13, 19, 20]. Image segmentation using rough sets has also been done [21, 22, 23, 24, 25].

In this paper, a hybrid algorithm called rough-fuzzy c-means (RFCM) algorithm is presented for segmentation of brain MR images. A preliminary version of this algorithm has been reported in [26, 27]. The RFCM algorithm is based on both rough sets and fuzzy sets. While the membership function of fuzzy sets enables efficient handling of overlapping partitions, the concept of lower and upper approximations of rough sets deals with uncertainty, vagueness, and incompleteness in class definition. Each partition is represented by a cluster prototype (centroid), a crisp lower approximation, and a fuzzy boundary. The lower approximation influences the fuzziness of the final partition. The cluster prototype (centroid) depends on the weighting average of the crisp lower approximation and fuzzy boundary. However, an important issue of the RFCM based brain MR image segmentation method is how to select initial prototypes of different classes or categories. The concept of discriminant analysis, based on the maximization of class separability, is used to circumvent the initialization and local minima problems of the RFCM, and enables efficient segmentation of brain MR images. The effectiveness of the RFCM algorithm, along with a comparison with other c-means algorithms, is demonstrated on a set of brain MR images using some standard validity indices.

The paper is organized as follows: Section 2 briefly introduces the necessary notions of fuzzy c-means and rough sets. In Section 3, the RFCM algorithm is described based on the theory of rough sets and fuzzy c-means. Section 4 gives an overview of the feature extraction techniques employed in segmentation of brain MR images along with the initialization method of c-means algorithm based on the maximization of class separability. Implementation details, experimental results, and a comparison among different c-means are presented in Section 5. Concluding remarks are given in Section 6.

2 Fuzzy C-Means and Rough Sets

This section presents the basic notions of fuzzy c-means and rough sets. The rough-fuzzy c-means (RFCM) algorithm is developed based on these algorithms.

2.1 Fuzzy C-Means

Let $X = \{x_1, \cdots, x_j, \cdots, x_n\}$ be the set of n objects and $V = \{v_1, \cdots, v_i, \cdots, v_c\}$ be the set of c centroids, where $x_j \in \Re^m$, $v_i \in \Re^m$, and $v_i \in X$. The fuzzy c-means provides a fuzzification of the hard c-means [9, 28]. It partitions X into c clusters by minimizing the objective function

$$J = \sum_{j=1}^{n} \sum_{i=1}^{c} (\mu_{ij})^{\acute{m}} ||x_j - v_i||^2 \tag{1}$$

where $1 \leq \acute{m} < \infty$ is the fuzzifier, v_i is the ith centroid corresponding to cluster β_i, $\mu_{ij} \in [0, 1]$ is the fuzzy membership of the pattern x_j to cluster β_i, and $||.||$ is the distance norm, such that

$$v_i = \frac{1}{n_i} \sum_{j=1}^{n} (\mu_{ij})^{\acute{m}} x_j; \text{ where } n_i = \sum_{j=1}^{n} (\mu_{ij})^{\acute{m}} \tag{2}$$

and

$$\mu_{ij} = (\sum_{k=1}^{c} (\frac{d_{ij}}{d_{kj}})^{\frac{2}{\acute{m}-1}})^{-1}; \text{ where } d_{ij}^2 = ||x_j - v_i||^2 \tag{3}$$

subject to

$$\sum_{i=1}^{c} \mu_{ij} = 1, \forall j, \text{ and } 0 < \sum_{j=1}^{n} \mu_{ij} < n, \forall i.$$

The process begins by randomly choosing c objects as the centroids (means) of the c clusters. The memberships are calculated based on the relative distance of the object x_j to the centroids by Equation 3. After computing memberships of all the objects, the new centroids of the clusters are calculated as per Equation 2. The process stops when the centroids stabilize. That is, the centroids from the previous iteration are identical to those generated in the current iteration. The basic steps are outlined as follows:

1. Assign initial means v_i, $i = 1, 2, \cdots, c$. Choose values for $\acute{m}$ and threshold ϵ. Set iteration counter $t = 1$.
2. Compute memberships μ_{ij} by Equation 3 for c clusters and n objects.
3. Update mean (centroid) v_i by Equation 2.
4. Repeat steps 2 to 4, by incrementing t, until $|\mu_{ij}(t) - \mu_{ij}(t-1)| > \epsilon$.

Although fuzzy c-means is a very useful clustering method, the resulting memberships values do not always correspond well to the degrees of belonging of the data, and it may be inaccurate in a noisy environment [29, 30]. In real data analysis, noise and outliers are unavoidable. Hence, to reduce this weakness of fuzzy c-means, and to produce memberships that have a good explanation of the degrees of belonging for the data, Krishnapuram and Keller [29, 30] proposed a possibilistic approach to clustering which used a possibilistic type of membership function to describe the degree of belonging. However, the possibilistic c-means sometimes generates coincident clusters [31]. Recently, the use of both fuzzy (probabilistic) and possibilistic memberships in a clustering algorithm has been proposed in [32].

2.2 Rough Sets

The theory of rough sets begins with the notion of an approximation space, which is a pair $< U, R >$, where U be a non-empty set (the universe of discourse) and R an equivalence relation on U, i.e., R is reflexive, symmetric, and transitive. The relation R decomposes the set U into disjoint classes in such a way that two elements x, y are in the same class iff $(x, y) \in R$. Let denote by U/R the quotient set of U by the relation R, and

$$U/R = \{X_1, X_2, \cdots, X_m\}$$

where X_i is an equivalence class of R, $i = 1, 2, \cdots, m$. If two elements $x, y \in U$ belong to the same equivalence class $X_i \in U/R$, then x and y are called indistinguishable. The equivalence classes of R and the empty set $\emptyset$ are the elementary sets in the approximation space $< U, R >$. Given an arbitrary set $X \in 2^U$, in general it may not be possible to describe X precisely in $< U, R >$. One may characterize X by a pair of lower and upper approximations defined as follows [15]:

$$\underline{R}(X) = \bigcup_{X_i \subseteq X} X_i; \quad \overline{R}(X) = \bigcup_{X_i \cap X \neq \emptyset} X_i$$

That is, the lower approximation $\underline{R}(X)$ is the union of all the elementary sets which are subsets of X, and the upper approximation $\overline{R}(X)$ is the union of all the elementary sets which have a non-empty intersection with X. The interval $[\underline{R}(X), \overline{R}(X)]$ is the representation of an ordinary set X in the approximation space $< U, R >$ or simply called the rough set of X. The lower (resp., upper) approximation $\underline{R}(X)$ (resp., $\overline{R}(X)$) is interpreted as the collection of those elements of U that definitely (resp., possibly) belong to X. Further,

– a set $X \in 2^U$ is said to be definable (or exact) in $< U, R >$ iff $\underline{R}(X) = \overline{R}(X)$.
– for any $X, Y \in 2^U$, X is said to be roughly included in Y, denoted by $X \tilde{\subset} Y$, iff $\underline{R}(X) \subseteq \underline{R}(Y)$ and $\overline{R}(X) \subseteq \overline{R}(Y)$.
– X and Y is said to be roughly equal, denoted by $X \simeq_R Y$, in $< U, R >$ iff $\underline{R}(X) = \underline{R}(Y)$ and $\overline{R}(X) = \overline{R}(Y)$.

In [15], Pawlak discusses two numerical characterizations of imprecision of a subset X in the approximation space $< U, R >$: accuracy and roughness. Accuracy of X, denoted by $\alpha_R(X)$, is simply the ratio of the number of objects in its lower approximation to that in its upper approximation; namely

$$\alpha_R(X) = \frac{|\underline{R}(X)|}{|\overline{R}(X)|}$$

The roughness of X, denoted by $\rho_R(X)$, is defined by subtracting the accuracy from 1:

$$\rho_R(X) = 1 - \alpha_R(X) = 1 - \frac{|\underline{R}(X)|}{|\overline{R}(X)|}$$

Note that the lower the roughness of a subset, the better is its approximation. Further, the following observations are easily obtained:

1. As $\underline{R}(X) \subseteq X \subseteq \overline{R}(X)$, $0 \leq \rho_R(X) \leq 1$.
2. By convention, when $X = \emptyset$, $\underline{R}(X) = \overline{R}(X) = \emptyset$ and $\rho_R(X) = 0$.
3. $\rho_R(X) = 0$ if and only if X is definable in $< U, R >$.

3 Rough-Fuzzy C-Means Algorithm

Incorporating both fuzzy and rough sets, next a newly introduced c-means algorithm, termed as rough-fuzzy c-means (RFCM) [26, 27], is described. The RFCM algorithm adds the concept of fuzzy membership of fuzzy sets, and lower and upper approximations of rough sets into c-means algorithm. While the membership of fuzzy sets enables efficient handling of overlapping partitions, the rough sets deal with uncertainty, vagueness, and incompleteness in class definition.

3.1 Objective Function

Let $\underline{A}(\beta_i)$ and $\overline{A}(\beta_i)$ be the lower and upper approximations of cluster β_i, and $B(\beta_i) = \{\overline{A}(\beta_i) - \underline{A}(\beta_i)\}$ denote the boundary region of cluster β_i. The RFCM partitions a set of n objects into c clusters by minimizing the objective function

$$J_{\mathrm{RF}} = \begin{cases} w \times \mathcal{A}_1 + \tilde{w} \times \mathcal{B}_1 & \text{if } \underline{A}(\beta_i) \neq \emptyset,\ B(\beta_i) \neq \emptyset \\ \mathcal{A}_1 & \text{if } \underline{A}(\beta_i) \neq \emptyset,\ B(\beta_i) = \emptyset \\ \mathcal{B}_1 & \text{if } \underline{A}(\beta_i) = \emptyset,\ B(\beta_i) \neq \emptyset \end{cases} \tag{4}$$

$$\mathcal{A}_1 = \sum_{i=1}^{c} \sum_{x_j \in \underline{A}(\beta_i)} ||x_j - v_i||^2 \qquad \mathcal{B}_1 = \sum_{i=1}^{c} \sum_{x_j \in B(\beta_i)} (\mu_{ij})^{\acute{m}} ||x_j - v_i||^2$$

v_i represents the centroid of the ith cluster β_i, the parameter w and $\tilde{w}$ correspond to the relative importance of lower bound and boundary region, and $w + \tilde{w} = 1$. Note that, μ_{ij} has the same meaning of membership as that in fuzzy c-means.

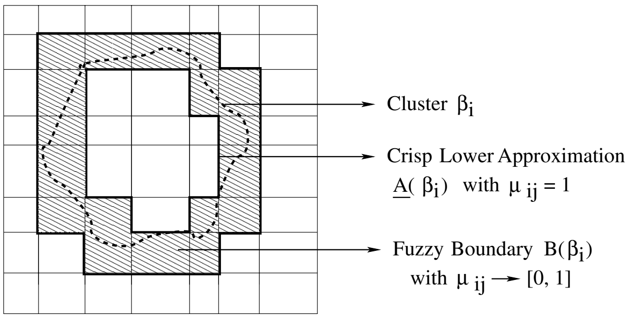


Fig. 1. RFCM: cluster β_i is represented by crisp lower bound and fuzzy boundary

In the RFCM, each cluster is represented by a centroid, a crisp lower approximation, and a fuzzy boundary (Fig. 1). The lower approximation influences the fuzziness of final partition. According to the definitions of lower approximations and boundary of rough sets, if an object $x_j \in \underline{A}(\beta_i)$, then $x_j \notin \underline{A}(\beta_k), \forall k \neq i$, and $x_j \notin B(\beta_i), \forall i$. That is, the object x_j is contained in β_i definitely. Thus, the weights of the objects in lower approximation of a cluster should be independent of other centroids and clusters, and should not be coupled with their similarity with respect to other centroids. Also, the objects in lower approximation of a cluster should have similar influence on the corresponding centroid and cluster. Whereas, if $x_j \in B(\beta_i)$, then the object x_j possibly belongs to β_i and potentially belongs to another cluster. Hence, the objects in boundary regions should have different influence on the centroids and clusters. So, in the RFCM, the membership values of objects in lower approximation are $\mu_{ij} = 1$, while those in boundary region are the same as fuzzy c-means (Equation 3). In other word, the RFCM algorithm first partitions the data into two classes - lower approximation and boundary. Only the objects in boundary are fuzzified.

3.2 Cluster Prototypes

The new centroid is calculated based on the weighting average of the crisp lower approximation and fuzzy boundary. Computation of the centroid is modified to include the effects of both fuzzy memberships and lower and upper bounds. The modified centroid calculation for the RFCM is obtained by solving Equation 4 with respect to v_i:

$$v_i^{\mathrm{RF}} = \begin{cases} w \times \mathcal{C}_1 + \tilde{w} \times \mathcal{D}_1 & \text{if } \underline{A}(\beta_i) \neq \emptyset,\ B(\beta_i) \neq \emptyset \\ \mathcal{C}_1 & \text{if } \underline{A}(\beta_i) \neq \emptyset,\ B(\beta_i) = \emptyset \\ \mathcal{D}_1 & \text{if } \underline{A}(\beta_i) = \emptyset,\ B(\beta_i) \neq \emptyset \end{cases} \tag{5}$$

$$\mathcal{C}_1 = \frac{1}{|\underline{A}(\beta_i)|} \sum_{x_j \in \underline{A}(\beta_i)} x_j; \quad \text{where } |\underline{A}(\beta_i)| \text{ represents the cardinality of } \underline{A}(\beta_i)$$

$$\text{and } \ \mathcal{D}_1 = \frac{1}{n_i} \sum_{x_j \in B(\beta_i)} (\mu_{ij})^{\acute{m}} x_j; \quad \text{where } \ n_i = \sum_{x_j \in B(\beta_i)} (\mu_{ij})^{\acute{m}}$$

Thus, the cluster prototypes (centroids) depend on the parameters w and $\tilde{w}$, and fuzzifier $\acute{m}$ rule their relative influence. The correlated influence of these parameters and fuzzifier, makes it somewhat difficult to determine their optimal values. Since the objects lying in lower approximation definitely belong to a cluster, they are assigned a higher weight w compared to $\tilde{w}$ of the objects lying in boundary region. Hence, for the RFCM, the values are given by $0 < \tilde{w} < w < 1$.

From the above discussions, the following properties of the RFCM algorithm can be derived.

1. $\bigcup \overline{A}(\beta_i) = U$, U be the set of objects of concern.
2. $\underline{A}(\beta_i) \cap \underline{A}(\beta_k) = \emptyset, \forall i \neq k$.
3. $\underline{A}(\beta_i) \cap B(\beta_i) = \emptyset, \forall i$.
4. $\exists i, k, B(\beta_i) \cap B(\beta_k) \neq \emptyset$.
5. $\mu_{ij} = 1, \forall x_j \in \underline{A}(\beta_i)$.
6. $\mu_{ij} \in [0, 1], \forall x_j \in B(\beta_i)$.

Let us briefly comment on some properties of the RFCM. The property 2 says that if an object $x_j \in \underline{A}(\beta_i) \Rightarrow x_j \notin \underline{A}(\beta_k), \forall k \neq i$. That is, the object x_j is contained in β_i definitely. The property 3 establishes the fact that if $x_j \in \underline{A}(\beta_i) \Rightarrow x_j \notin B(\beta_i)$, - that is, an object may not be in both lower and boundary region of a cluster β_i. The property 4 says that if $x_j \in B(\beta_i) \Rightarrow \exists k, x_j \in B(\beta_k)$. It means an object $x_j \in B(\beta_i)$ possibly belongs to β_i and potentially belongs to other cluster. The properties 5 and 6 are of great importance in computing the objective function J_{RF} and the cluster prototype v^{RF}. They say that the membership values of the objects in lower approximation are $\mu_{ij} = 1$, while those in boundary region are the same as fuzzy c-means. That is, each cluster β_i consists of a crisp lower approximation $\underline{A}(\beta_i)$ and a fuzzy boundary $B(\beta_i)$.

3.3 Details of the Algorithm

Approximate optimization of J_{RF} (Equation 4) by the RFCM is based on Picard iteration through Equations 3 and 5. This type of iteration is called alternating optimization. The process starts by randomly choosing c objects as the centroids of the c clusters. The fuzzy memberships of all objects are calculated using Equation 3.

Let $\mu_i = (\mu_{i1}, \cdots, \mu_{ij}, \cdots, \mu_{in})$ represent the fuzzy cluster β_i associated with the centroid v_i. After computing μ_{ij} for c clusters and n objects, the values of μ_{ij} for each object x_j are sorted and the difference of two highest memberships of x_j is compared with a threshold value δ. Let μ_{ij} and μ_{kj} be the highest and second highest memberships of x_j. If $(\mu_{ij} - \mu_{kj}) > \delta$, then $x_j \in \underline{A}(\beta_i)$ as well as $x_j \in \overline{A}(\beta_i)$, otherwise $x_j \in \overline{A}(\beta_i)$ and $x_j \in \overline{A}(\beta_k)$. After assigning each object in lower approximations or boundary regions of different clusters based on δ, memberships μ_{ij} of the objects are modified. The values of μ_{ij} are set to 1 for the objects in lower approximations, while those in boundary regions are remain unchanged. The new centroids of the clusters are calculated as per Equation 5. The main steps of the RFCM algorithm proceed as follows:

1. Assign initial centroids v_i, $i = 1, 2, \cdots, c$. Choose values for fuzzifier $\acute{m}$, and thresholds ϵ and δ. Set iteration counter $t = 1$.
2. Compute μ_{ij} by Equation 3 for c clusters and n objects.
3. If μ_{ij} and μ_{kj} be the two highest memberships of x_j and $(\mu_{ij} - \mu_{kj}) \leq \delta$, then $x_j \in \overline{A}(\beta_i)$ and $x_j \in \overline{A}(\beta_k)$. Furthermore, x_j is not part of any lower bound.
4. Otherwise, $x_j \in \underline{A}(\beta_i)$. In addition, by properties of rough sets, $x_j \in \overline{A}(\beta_i)$.
5. Modify μ_{ij} considering lower and boundary regions for c clusters and n objects.
6. Compute new centroid as per Equation 5.
7. Repeat steps 2 to 7, by incrementing t, until $|\mu_{ij}(t) - \mu_{ij}(t-1)| > \epsilon$.

The performance of the RFCM depends on the value of δ, which determines the class labels of all the objects. In other word, the RFCM partitions the data set into two classes - lower approximation and boundary, based on the value of δ. In the present work, the following definition is used:

$$\delta = \frac{1}{n}\sum_{j=1}^{n}(\mu_{ij} - \mu_{kj}) \tag{6}$$

where n is the total number of objects, μ_{ij} and μ_{kj} are the highest and second highest memberships of x_j. That is, the value of δ represents the average difference of two highest memberships of all the objects in the data set. A good clustering procedure should make the value of δ as high as possible. The value of δ is, therefore, data dependent.

4 Segmentation of Brain MR Images

In this section, the feature extraction methodology for segmentation of brain MR images is first described. Next, the methodology to select initial centroids for different c-means algorithms is provided based on the concept of maximization of class separability.

4.1 Feature Extraction

Statistical texture analysis derives a set of statistics from the distribution of pixel values or blocks of pixel values. There are different types of statistical texture, first-order, second-order, and higher order statistics, based on the number of pixel combinations used to compute the textures. The first-order statistics, like mean, standard deviation, range, entropy, and the qth moment about the mean, are calculated using the histogram formed by the gray scale value of each pixel. These statistics consider the properties of the gray scale values, but not their spatial distribution. The second-order statistics are based on pairs of pixels. This takes into account the spatial distribution of the gray scale distribution. In the present work, only first- and second-order statistical textures are considered.

A set of 13 input features is used for clustering the brain MR images. These include gray value of the pixel, two proposed features (first order statistics) - homogeneity and edge value of the pixel, and 10 Haralick's textural features [33] (second order statistics) - angular second moment, contrast, correlation, inverse difference moment, sum average, sum variance, sum entropy, second order entropy, difference variance, and difference entropy. They are useful in characterizing images, and can be used as features of a pixel. Hence these features have promising application in clustering based brain MRI segmentation.

Homogeneity. If H is the homogeneity of a pixel $\mathrm{I}_{m,n}$ within 3×3 neighborhood, then

$$\mathrm{H} = 1 - \frac{1}{6(\mathrm{I}_{\max} - \mathrm{I}_{\min})}\{|\mathrm{I}_{m-1,n-1} + \mathrm{I}_{m+1,n+1} - \mathrm{I}_{m-1,n+1} - \mathrm{I}_{m+1,n-1}| + \\ |\mathrm{I}_{m-1,n-1} + 2\mathrm{I}_{m,n-1} + \mathrm{I}_{m+1,n-1} - \mathrm{I}_{m-1,n+1} - 2\mathrm{I}_{m,n+1} - \mathrm{I}_{m+1,n+1}|\}$$

where $\mathrm{I}_{\max}$ and $\mathrm{I}_{\min}$ represent the maximum and minimum gray values of the image. The region that is entirely within an organ will have a high H value. On the other hand, the regions that contain more than one organ will have lower H values.

Edge Value. In MR imaging, the histogram of the given image is in general unimodal. One side of the peak may display a shoulder or slope change, or one side may be less steep than the other, reflecting the presence of two peaks that are close together or that differ greatly in height. The histogram may also contain a third, usually smaller, population corresponding to points on the object-background border. These points have gray levels intermediate between those of the object and background; their presence raises the level of the valley floor between the two peaks, or if the peaks are already close together, makes it harder to detect the fact that they are not a single peak.

As the histogram peaks are close together and very unequal in size, it may be difficult to detect the valley between them. In determining how each point of the image should contribute to the segmentation method, the proposed method takes into account the rate of change of gray level at the point, as well as the point's gray level (edge value); that is, the maximum of differences of average gray levels in pairs of horizontally and vertically adjacent 2×2 neighborhoods [4, 34]. If $\varDelta$ is the edge value at a given point $\mathrm{I}_{m,n}$, then

$$\begin{aligned}\varDelta = \frac{1}{4}\max\{|\mathrm{I}_{m-1,n} + \mathrm{I}_{m-1,n+1} + \mathrm{I}_{m,n} + \mathrm{I}_{m,n+1} - \mathrm{I}_{m+1,n} - \mathrm{I}_{m+1,n+1}\\ -\mathrm{I}_{m+2,n} - \mathrm{I}_{m+2,n+1}|, |\mathrm{I}_{m,n-1} + \mathrm{I}_{m,n} + \mathrm{I}_{m+1,n-1}\\ +\mathrm{I}_{m+1,n} - \mathrm{I}_{m,n+1} - \mathrm{I}_{m,n+2} - \mathrm{I}_{m+1,n+1} - \mathrm{I}_{m+1,n+2}|\}\end{aligned}$$

According to the image model, points interior to the object and background should generally have low edge values, since they are highly correlated with their neighbors, while those on the object-background border should have high edge values [4].

Haralick's Textural Feature. Texture is one of the important features used in identifying objects or regions of interest in an image. It is often described as a set of statistical measures of the spatial distribution of gray levels in an image. This scheme has been found to provide a powerful input feature representation for various recognition problems. Haralick et al. [33] proposed different textural properties for image classification. Haralick's textural measures are based upon the moments of a joint probability density function that is estimated as the joint co-occurrence matrix or gray level co-occurrence matrix [33, 35]. It reflects the distribution of the probability of occurrence of a pair of gray levels separated by a given distance d at angle θ. Based upon normalized gray level co-occurrence matrix, Haralick proposed several quantities as measure of texture like energy, contrast, correlation, sum of squares, inverse difference moments, sum average, sum variance, sum entropy, entropy, difference variance, difference entropy, information measure of correlation 1, and correlation 2. In [33], these properties were calculated for large blocks in aerial photographs. Every pixel within these each large block was then assigned the same texture values. This leads to a significant loss of resolution that is unacceptable in medical imaging.

In the present work, the texture values are assigned to a pixel by using a 3×3 sliding window centered about that pixel. The gray level co-occurrence matrix is constructed by mapping the gray level co-occurrence probabilities based on spatial relations of pixels in different angular directions ($\theta = 0°, 45°, 90°, 135°$) with unit pixel distance, while scanning the window (centered about a pixel) from left-to-right and top-to-bottom [33, 35]. Ten texture measures - angular second moment, contrast, correlation, inverse difference moment, sum average, sum variance, sum entropy, second order entropy, difference variance, and difference entropy, are computed for each window. For four angular directions, a set of four values is obtained for each of ten measures. The mean of each of the ten measures, averaged over four values, along with gray value, homogeneity, and edge value of the pixel, comprise the set of 13 features which is used as feature vector of the corresponding pixel.

4.2 Selection of Initial Centroids

A limitation of the c-means algorithm is that it can only achieve a local optimum solution that depends on the initial choice of the centroids. Consequently, computing resources may be wasted in that some initial centroids get stuck in regions of the input space with a scarcity of data points and may therefore never have the chance to move to new locations where they are needed. To overcome this limitation of the *c*-means algorithm, next a method is described to select initial centroids, which is based on discriminant analysis maximizing some measures of class separability [36]. It enables the algorithm to converge to an optimum or near optimum solutions.

Prior to describe the proposed method for selecting initial centroids, next a quantitative measure of class separability [36] is provided that is given by

$$\mathrm{J(T)} = \frac{\mathrm{P_1(T)P_2(T)[m_1(T) - m_2(T)]^2}}{\mathrm{P_1(T)}\sigma_1^2\mathrm{(T)} + \mathrm{P_2(T)}\sigma_2^2\mathrm{(T)}} \tag{7}$$

where

$$\mathrm{P_1(T)} = \sum_{z=0}^{T} \mathrm{h(z)}; \quad \mathrm{P_2(T)} = \sum_{z=T+1}^{L-1} \mathrm{h(z)} = 1 - \mathrm{P_1(T)}$$

$$\mathrm{m_1(T)} = \frac{1}{\mathrm{P_1(T)}} \sum_{z=0}^{T} \mathrm{zh(z)}; \quad \mathrm{m_2(T)} = \frac{1}{\mathrm{P_2(T)}} \sum_{z=T+1}^{L-1} \mathrm{zh(z)}$$

$$\sigma_1^2\mathrm{(T)} = \frac{1}{\mathrm{P_1(T)}} \sum_{z=0}^{T} [\mathrm{z - m_1(T)}]^2 \mathrm{h(z)}; \quad \sigma_2^2\mathrm{(T)} = \frac{1}{\mathrm{P_2(T)}} \sum_{z=T+1}^{L-1} [\mathrm{z - m_2(T)}]^2 \mathrm{h(z)}$$

Here, L is the total number of discrete values ranging between $[0, \mathrm{L} - 1]$, T is the threshold value, which maximizes J(T), and h(z) represents the percentage of data having feature value z over the total number of discrete values of the corresponding feature. To maximize J(T), the means of the two classes should be as well separated as possible and the variances in both classes should be as small as possible.

Based on the concept of maximization of class separability, the method for selecting initial centroids is described next. The main steps of this method proceeds as follows.

1. The data set $X = \{x_1, \cdots, x_j, \cdots, x_n\}$ with $x_j \in \Re^m$ are first discretized to facilitate class separation method. Suppose, the possible value range of a feature $\mathrm{f_m}$ in the data set is $(\mathrm{f_{m,min}}, \mathrm{f_{m,max}})$, and the real value that the data element x_j takes at $\mathrm{f_m}$ is $\mathrm{f_{mj}}$, then the discretized value of $\mathrm{f_{mj}}$ is

$$\mathrm{Discretized(f_{mj})} = (\mathrm{L} - 1) \times \left\{ \frac{\mathrm{f_{mj} - f_{m,min}}}{\mathrm{f_{m,max} - f_{m,min}}} \right\} \tag{8}$$

   where L is the total number of discrete values ranging between $[0, \mathrm{L} - 1]$.
2. For each feature $\mathrm{f_m}$, calculate h(z) for $0 \leq \mathrm{z} < \mathrm{L}$.
3. Calculate the threshold value $\mathrm{T_m}$ for the feature $\mathrm{f_m}$, which maximizes class separability along that feature.

4. Based on the threshold $\mathrm{T_m}$, discretize the corresponding feature $\mathrm{f_m}$ of the data element x_j as follows

$$\overline{\mathrm{f}}_{\mathrm{mj}} = \begin{cases} 1, \text{if Discretized}(\mathrm{f_{mj}}) \geq \mathrm{T_m} \\ 0, \text{Otherwise} \end{cases}$$

5. Repeat steps 2 to 4 for all the features and generate the set of discretized objects $\overline{X} = \{\overline{x}_1, \cdots, \overline{x}_j, \cdots, \overline{x}_n\}$.
6. Calculate total number of similar discretized objects $\mathrm{N}(x_i)$ and mean of similar objects $\overline{v}(x_i)$ of x_i as

$$\mathrm{N}(x_i) = \sum_{j=1}^{n} \delta_j \quad \text{and} \quad \overline{v}(x_i) = \frac{1}{\mathrm{N}(x_i)} \sum_{j=1}^{n} \delta_j \times x_j$$

$$\text{where} \quad \delta_j = \begin{cases} 1 \text{ if } \overline{x}_j = \overline{x}_i \\ 0 \text{ Otherwise} \end{cases}$$

7. Sort n objects according to their values of $\mathrm{N}(x_i)$ such that $\mathrm{N}(x_1) > \mathrm{N}(x_2) > \cdots > \mathrm{N}(x_n)$.
8. If $\overline{x}_i = \overline{x}_j$, then $\mathrm{N}(x_i) = \mathrm{N}(x_j)$ and $\overline{v}(x_j)$ should not be considered as a centroid (mean), resulting in a reduced set of objects to be considered for initial centroids.
9. Let there be $\acute{n}$ objects in the reduced set having $\mathrm{N}(x_i)$ values such that $\mathrm{N}(x_1) > \mathrm{N}(x_2) > \cdots > \mathrm{N}(x_{\acute{n}})$. A heuristic threshold function can be defined as follows [37]:

$$\mathrm{Tr} = \frac{\mathrm{R}}{\tilde{\epsilon}}; \quad \text{where } \mathrm{R} = \sum_{i=1}^{\acute{n}} \frac{1}{\mathrm{N}(x_i) - \mathrm{N}(x_{i+1})}$$

where $\tilde{\epsilon}$ is a constant ($= 0.5$, say), so that all the means $\overline{v}(x_i)$ of the objects in reduced set having $\mathrm{N}(x_i)$ value higher than it are regarded as the candidates for initial centroids (means).

The value of Tr is high if most of the $\mathrm{N}(x_i)$'s are large and close to each other. The above condition occurs when a small number of large clusters are present. On the other hand, if the $\mathrm{N}(x_i)$'s have wide variation among them, then the number of clusters with smaller size increases. Accordingly, Tr attains a lower value automatically. Note that the main motive of introducing this threshold function lies in reducing the number of centroids. Actually, it attempts to eliminate noisy centroids (data representatives having lower values of $\mathrm{N}(x_i)$) from the whole data set. The whole approach is, therefore, data dependent.

5 Performance Analysis

In this section, the performance of different c-means algorithms on segmentation of brain MR images is presented. Above 100 MR images with different sizes and 16 bit gray levels are tested using different c-means. All the brain MR images are collected from the Advanced Medicare and Research Institute, Kolkata, India. The performance of the RFCM is compared extensively with that of different c-means algorithms. These involve different combinations of the individual components of the hybrid scheme. The algorithms compared are hard c-means (HCM), fuzzy c-means (FCM) [9, 28], possibilistic c-means (PCM) [29, 30], fuzzy-possibilistic c-means (FPCM) [32], and rough c-means (RCM) [38]. All algorithms are implemented in C language and run in LINUX platform having machine configuration Pentium IV, 3.2 GHz, 1 MB cache, and 1 GB RAM.

5.1 Quantitative Indices

The comparative performance of different c-means is reported with respect to DB and Dunn index [39], and β index [40], which are described next.

Davies-Bouldin (DB) Index: The Davies-Bouldin (DB) index [39] is a function of the ratio of sum of within-cluster distance to between-cluster separation and is given by

$$\mathrm{DB} = \frac{1}{c}\sum_{i=1}^{c} \max_{i \neq k}\left\{\frac{S(v_i)+S(v_k)}{d(v_i, v_k)}\right\}$$

for $1 \leq i, k \leq c$. The DB index minimizes the within-cluster distance $S(v_i)$ and maximizes the between-cluster separation $d(v_i, v_k)$. Therefore, for a given data set and c value, the higher the similarity values within the clusters and the between-cluster separation, the lower would be the DB index value. A good clustering procedure should make the value of DB index as low as possible.

Dunn Index: Dunn index [39] is also designed to identify sets of clusters that are compact and well separated. Dunn index maximizes

$$\mathrm{Dunn} = \min_i\left\{\min_{i \neq k}\left\{\frac{d(v_i, v_k)}{\max_l S(v_l)}\right\}\right\} \qquad \text{for } 1 \leq i, k, l \leq c.$$

β Index: The β-index of Pal et al. [40] is defined as the ratio of the total variation and within-cluster variation, and is given by

$$\beta = \frac{\mathrm{N}}{\mathrm{M}}; \text{ where } \mathrm{N} = \sum_{i=1}^{c}\sum_{j=1}^{n_i} ||x_{ij} - \overline{v}||^2; \mathrm{M} = \sum_{i=1}^{c}\sum_{j=1}^{n_i} ||x_{ij} - v_i||^2; \sum_{i=1}^{c} n_i = n;$$

n_i is the number of objects in the ith cluster ($i = 1, 2, \cdots, c$), n is the total number of objects, x_{ij} is the jth object in cluster i, v_i is the mean or centroid of ith cluster, and $\overline{v}$ is the mean of n objects. For a given image and c value, the higher the homogeneity within the segmented regions, the higher would be the β value. The value of β also increases with c.

5.2 Example

Consider Fig. 2 as an example that represents an MR image (I-20497774) along with the segmented images obtained using different c-means algorithms. Each image is of size 256×180 with 16 bit gray levels. So, the number of objects in the data set of I-20497774 is 46080. The parameters generated in the proposed initialization method are shown in Table 1 only for I-20497774 data set along with the values of input parameters. The threshold values for 13 features of the given data set are also reported in this table. Table 2 depicts the values of DB index, Dunn index, and β index of FCM and RFCM for different values of c on the data set of I-20497774, considering $w = 0.95$ and $\acute{m} = 2.0$. The results reported here with respect to DB and Dunn index confirm that both FCM and RFCM achieve their best results for $c = 4$. Also, the value of β index, as expected, increases with increase in the value of c. For a particular value of c, the performance of RFCM is better than that of FCM.

Finally, Table 3 provides the comparative results of different c-means algorithms on I-20497774 with respect to the values of DB index, Dunn index, and β index. The corresponding segmented images along with the original one are presented in Fig. 2. The results reported in Fig. 2 and Table 3 confirm that the RFCM algorithm produces segmented image more promising than do the

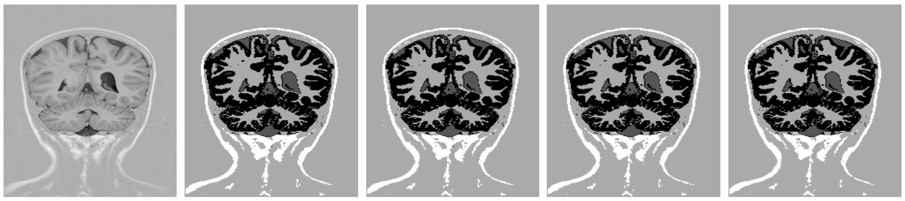

(a) Original (b) HCM (c) FCM (d) RCM (e) RFCM

Fig. 2. I-20497774: original and segmented images of different c-means

Table 1. Values of Different Parameters

Size of image = 256×180
Minimum gray value = 1606, Maximum gray value = 2246
Samples per pixel = 1, Bits allocated = 16, Bits stored = 12

Number of objects = 46080
Number of features = 13, Value of L = 101

Threshold Values:
Gray value = 1959, Homogeneity = 0.17, Edge value = 0.37
Angular second moment = 0.06, Contrast = 0.12
Correlation = 0.57, Inverse difference moment = 0.18
Sum average = 0.17, Sum variance = 0.14, Sum entropy = 0.87
Entropy = 0.88, Difference variance = 0.07, Difference entropy = 0.79

Table 2. Performance of FCM and RFCM on I-20497774 data set

Value	DB Index		Dunn Index		β Index	
of c	FCM	RFCM	FCM	RFCM	FCM	RFCM
2	0.38	0.19	2.17	3.43	3.62	4.23
3	0.22	0.16	1.20	1.78	7.04	7.64
4	0.15	0.13	1.54	1.80	11.16	13.01
5	0.29	0.19	0.95	1.04	11.88	14.83
6	0.24	0.23	0.98	1.11	19.15	19.59
7	0.23	0.21	1.07	0.86	24.07	27.80
8	0.31	0.21	0.46	0.95	29.00	33.02
9	0.30	0.24	0.73	0.74	35.06	40.07
10	0.30	0.22	0.81	0.29	41.12	44.27

Table 3. Performance of Different C-Means on I-20497774 data set

Algorithms	DB Index	Dunn Index	β Index
HCM	0.17	1.28	10.57
FCM	0.15	1.54	11.16
RCM	0.16	1.56	11.19
RFCM	0.13	1.80	13.01

conventional c-means algorithms. Some of the existing algorithms like PCM and FPCM fail to produce multiple segments as they generate coincident clusters even when they are initialized with final prototypes of the FCM.

5.3 Haralick's Features Versus Proposed Features

Table 4 presents the comparative results of different c-means for proposed and Haralick's features on I-20497774 data set. While P-2 and H-13 stand for the set of two proposed features and thirteen Haralick's features, H-10 represents that of ten Haralick's features which are used in the current study. The proposed features are found as important as Haralick's ten features for clustering based segmentation of brain MR images. The set of 13 features, comprising of gray value, two proposed features, and ten Haralick's features, improves the performance of all c-means with respect to DB, Dunn, and β. It is also observed that the Haralick's three features - sum of squares, information measure of correlation 1, and correlation 2, do not contribute any extra information for segmentation of brain MR images.

5.4 Random Versus Proposed Initialization Method

Table 5 provides comparative results of different c-means algorithms with random initialization of centroids and the proposed discriminant analysis based initialization method described in Section 4.2 for the data sets I-20497761, I-20497763, and I-20497777 (Fig. 3). The proposed initialization method is found

Table 4. Haralick's and Proposed Features on I-20497774 data set

Algorithms	Features	DB Index	Dunn Index	β Index	Time (ms)
HCM	H-13	0.19	1.28	10.57	4308
	H-10	0.19	1.28	10.57	3845
	P-2	0.18	1.28	10.57	1867
	H-10 $\cup$ P-2	0.17	1.28	10.57	3882
FCM	H-13	0.15	1.51	10.84	36711
	H-10	0.15	1.51	10.84	34251
	P-2	0.15	1.51	11.03	14622
	H-10 $\cup$ P-2	0.15	1.54	11.16	43109
RCM	H-13	0.19	1.52	11.12	5204
	H-10	0.19	1.52	11.12	5012
	P-2	0.17	1.51	11.02	1497
	H-10 $\cup$ P-2	0.16	1.56	11.19	7618
RFCM	H-13	0.13	1.76	12.57	15705
	H-10	0.13	1.76	12.57	15414
	P-2	0.13	1.77	12.88	6866
	H-10 $\cup$ P-2	0.13	1.80	13.01	17084

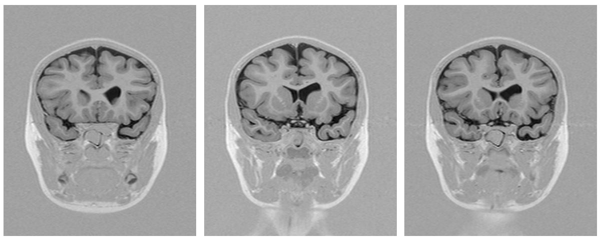

(a) I-20497761 (b) I-20497763 (c) I-20497777

Fig. 3. Examples of some brain MR images

to improve the performance in terms of DB index, Dunn index, and β index as well as reduce the time requirement of all c-means algorithms. It is also observed that HCM with proposed initialization method performs similar to RFCM with random initialization, although it is expected that RFCM is superior to HCM in partitioning the objects. While in random initialization, the c-means algorithms get stuck in local optimums, the proposed initialization method enables the algorithms to converge to an optimum or near optimum solutions. In effect, the execution time required for different c-means algorithms is lesser in proposed scheme compared to random initialization.

5.5 Comparative Performance Analysis

Table 6 compares the performance of different c-means algorithms on some brain MR images with respect to DB, Dunn, and β index. The original images along with the segmented versions of different c-means are shown in Figs. 4-6. All

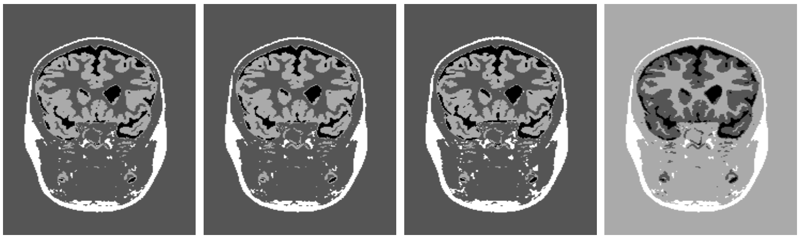
Table 5. Performance of Random and Proposed Initialization Method

Data Set	Algorithms	Initialization	DB Index	Dunn Index	β Index	Time (ms)
I-204 97761	HCM	Random	0.23	1.58	9.86	8297
		Proposed	0.15	2.64	12.44	4080
	FCM	Random	0.19	1.63	12.73	40943
		Proposed	0.12	2.69	13.35	38625
	RCM	Random	0.19	1.66	10.90	9074
		Proposed	0.14	2.79	12.13	6670
	RFCM	Random	0.15	2.07	11.89	19679
		Proposed	0.11	2.98	13.57	16532
I-204 97763	HCM	Random	0.26	1.37	10.16	3287
		Proposed	0.16	2.03	13.18	3262
	FCM	Random	0.21	1.54	10.57	46157
		Proposed	0.15	2.24	13.79	45966
	RCM	Random	0.21	1.60	10.84	10166
		Proposed	0.14	2.39	13.80	6770
	RFCM	Random	0.17	1.89	11.49	19448
		Proposed	0.10	2.38	14.27	15457
I-204 97777	HCM	Random	0.33	1.52	6.79	4322
		Proposed	0.16	2.38	8.94	3825
	FCM	Random	0.28	1.67	7.33	42284
		Proposed	0.15	2.54	10.02	40827
	RCM	Random	0.27	1.71	7.47	8353
		Proposed	0.13	2.79	9.89	7512
	RFCM	Random	0.19	1.98	8.13	18968
		Proposed	0.11	2.83	11.04	16930

Table 6. Performance of Different C-Means Algorithms

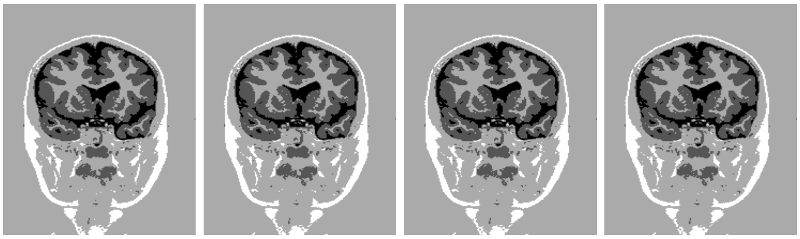
Data Set	Algorithms	DB Index	Dunn Index	β Index	Time (ms)
I-204 97761	HCM	0.15	2.64	12.44	4080
	FCM	0.12	2.69	13.35	38625
	RCM	0.14	2.79	12.13	6670
	RFCM	0.11	2.98	13.57	16532
I-204 97763	HCM	0.16	2.03	13.18	3262
	FCM	0.15	2.24	13.79	45966
	RCM	0.14	2.39	13.80	6770
	RFCM	0.10	2.38	14.27	15457
I-204 97777	HCM	0.16	2.38	8.94	3825
	FCM	0.15	2.54	10.02	40827
	RCM	0.13	2.79	9.89	7512
	RFCM	0.11	2.83	11.04	16930

the results reported in Table 6 and Figs. 4-6 confirm that although each *c*-means algorithm, except PCM and FPCM, generates good segmented images, the values of DB, Dunn, and β index of the RFCM are better compared to other



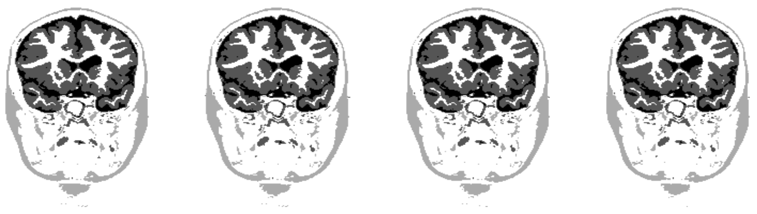
(a) HCM (b) FCM (c) RCM (d) RFCM

Fig. 4. I-20497761: segmented versions of different c-means algorithms



(a) HCM (b) FCM (c) RCM (d) RFCM

Fig. 5. I-20497763: segmented versions of different c-means algorithms



(a) HCM (b) FCM (c) RCM (d) RFCM

Fig. 6. I-20497777: segmented versions of different c-means algorithms

c-means algorithms. Both PCM and FPCM fail to produce multiple segments of the brain MR images as they generate coincident clusters even when they are initialized with the final prototypes of other c-means algorithms.

Table 6 also provides execution time (in milli sec.) of different c-means. The execution time required for the RFCM is significantly lesser compared to FCM. For the HCM and RCM, although the execution time is less, the performance is considerably poorer than that of RFCM. Following conclusions can be drawn from the results reported in this paper:

1. It is observed that RFCM is superior to other c-means algorithms. However, RFCM requires higher time compared to HCM/RCM and lesser time compared to FCM. But, the performance of RFCM with respect to DB, Dunn,

and β is significantly better than all other c-means. The performance of FCM and RCM is intermediate between RFCM and HCM.
2. The discriminant analysis based initialization is found to improve the values of DB, Dunn, and β as well as reduce the time requirement substantially for all c-means algorithms.
3. The proposed two features are as important as Haralick's ten features for clustering based segmentation of brain MR images.
4. Use of rough sets and fuzzy memberships adds a small computational load to HCM algorithm; however the corresponding integrated method (RFCM) shows a definite increase in Dunn index and decrease in DB index.

The best performance of the proposed method in terms of DB, Dunn, and β is achieved due to the following reasons:

1. the discriminant analysis based initialization of centroids enables the algorithm to converge to an optimum or near optimum solutions;
2. membership of the RFCM handles efficiently overlapping partitions; and
3. the concept of crisp lower bound and fuzzy boundary of the RFCM algorithm deals with uncertainty, vagueness, and incompleteness in class definition.

In effect, promising segmented brain MR images are obtained using the RFCM algorithm.

6 Conclusion and Future Works

A robust segmentation technique is presented in this paper, integrating the merits of rough sets, fuzzy sets, and c-means algorithm, for brain MR images. Some new measures are introduced, based on the local properties of MR images, for accurate segmentation. The method, based on the concept of maximization of class separability, is found to be successful in effectively circumventing the initialization and local minima problems of iterative refinement clustering algorithms like c-means. The effectiveness of the algorithm, along with a comparison with other algorithms, is demonstrated on a set of brain MR images. The extensive experimental results show that the rough-fuzzy c-means algorithm produces a segmented image more promising than do the conventional algorithms.

Although the proposed methodology of integrating rough sets, fuzzy sets, and c-means algorithm is efficiently demonstrated for segmentation of brain MR images, the concept can be applied to other unsupervised classification problems. An MR image based epilepsy diagnosis system is being developed by the authors, and this was the initial motivation to develop segmentation method, since segmentation is a key stage in successful diagnosis.

Acknowledgments. The authors would like to thank the Department of Science and Technology, Government of India, for funding the Center for Soft Computing Research: A National Facility under its IRHPA scheme and the Advanced Medicare and Research Institute, Kolkata, India, for providing brain MR images. The work was done when one of the authors, S. K. Pal, was a J. C. Bose Fellow of the Government of India.

References

1. Rosenfeld, A., Kak, A.C.: 10. In: Digital Picture Processing. Academic Press, Inc., London (1982)
2. Suetens, P.: 6. In: Fundamentals of Medical Imaging. Cambridge University Press, Cambridge (2002)
3. Lee, C., Hun, S., Ketter, T.A., Unser, M.: Unsupervised Connectivity Based Thresholding Segmentation of Midsaggital Brain MR Images. Computers in Biology and Medicine 28, 309–338 (1998)
4. Maji, P., Kundu, M.K., Chanda, B.: Second Order Fuzzy Measure and Weighted Co-Occurrence Matrix for Segmentation of Brain MR Images. Fundamenta Informaticae, 1–15 (2008)
5. Manousakes, I.N., Undrill, P.E., Cameron, G.G.: Split and Merge Segmentation of Magnetic Resonance Medical Images: Performance Evaluation and Extension to Three Dimensions. Computers and Biomedical Research 31, 393–412 (1998)
6. Singleton, H.R., Pohost, G.M.: Automatic Cardiac MR Image Segmentation Using Edge Detection by Tissue Classification in Pixel Neighborhoods. Magnetic Resonance in Medicine 37, 418–424 (1997)
7. Pal, N.R., Pal, S.K.: A Review on Image Segmentation Techniques. Pattern Recognition 26, 1277–1294 (1993)
8. Rajapakse, J.C., Giedd, J.N., Rapoport, J.L.: Statistical Approach to Segmentation of Single Channel Cerebral MR Images. IEEE Transactions on Medical Imaging 16, 176–186 (1997)
9. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithm. Plenum, New York (1981)
10. Leemput, K.V., Maes, F., Vandermeulen, D., Suetens, P.: Automated Model-Based Tissue Classification of MR Images of the Brain. IEEE Transactions on Medical Imaging 18, 897–908 (1999)
11. Wells III, W.M., Grimson, W.E.L., Kikinis, R., Jolesz, F.A.: Adaptive Segmentation of MRI Data. IEEE Transactions on Medical Imaging 15, 429–442 (1996)
12. Cagnoni, S., Coppini, G., Rucci, M., Caramella, D., Valli, G.: Neural Network Segmentation of Magnetic Resonance Spin Echo Images of the Brain. Journal of Biomedical Engineering 15, 355–362 (1993)
13. Hall, L.O., Bensaid, A.M., Clarke, L.P., Velthuizen, R.P., Silbiger, M.S., Bezdek, J.C.: A Comparison of Neural Network and Fuzzy Clustering Techniques in Segmenting Magnetic Resonance Images of the Brain. IEEE Transactions on Neural Network 3, 672–682 (1992)
14. Zadeh, L.A.: Fuzzy Sets. Information and Control 8, 338–353 (1965)
15. Pawlak, Z.: Rough Sets, Theoretical Aspects of Resoning About Data. Kluwer, Dordrecht (1991)
16. Dubois, D., Prade, H.: Rough Fuzzy Sets and Fuzzy Rough Sets. International Journal of General Systems 17, 191–209 (1990)
17. Maji, P., Pal, S.K.: Rough-Fuzzy C-Medoids Algorithm and Selection of Bio-Basis for Amino Acid Sequence Analysis. IEEE Transactions on Knowledge and Data Engineering 19, 859–872 (2007)
18. Pal, S.K., Mitra, S., Mitra, P.: Rough-Fuzzy MLP: Modular Evolution, Rule Generation, and Evaluation. IEEE Transactions on Knowledge and Data Engineering 15, 14–25 (2003)
19. Brandt, M.E., Bohan, T.P., Kramer, L.A., Fletcher, J.M.: Estimation of CSF, White and Gray Matter Volumes in Hydrocephalic Children Using Fuzzy Clustering of MR Images. Computerized Medical Imaging and Graphics 18, 25–34 (1994)

20. Li, C.L., Goldgof, D.B., Hall, L.O.: Knowledge-Based Classification and Tissue Labeling of MR Images of Human Brain. IEEE Transactions on Medical Imaging 12, 740–750 (1993)
21. Mushrif, M.M., Ray, A.K.: Color Image Segmentation: Rough-Set Theoretic Approach. Pattern Recognition Letters 29, 483–493 (2008)
22. Pal, S.K., Mitra, P.: Multispectral Image Segmentation Using Rough Set Initiatized EM Algorithm. IEEE Transactions on Geoscience and Remote Sensing 40, 2495–2501 (2002)
23. Widz, S., Revett, K., Slezak, D.: A Hybrid Approach to MR Imaging Segmentation Using Unsupervised Clustering and Approximate Reducts. In: Proceedings of the 10th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, pp. 372–382 (2005)
24. Widz, S., Revett, K., Slezak, D.: A Rough Set-Based Magnetic Resonance Imaging Partial Volume Detection System. In: Proceedings of the First International Conference on Pattern Recognition and Machine Intelligence, pp. 756–761 (2005)
25. Widz, S., Slezak, D.: Approximation Degrees in Decision Reduct-Based MRI Segmentation. In: Proceedings of the Frontiers in the Convergence of Bioscience and Information Technologies, pp. 431–436 (2007)
26. Maji, P., Pal, S.K.: RFCM: A Hybrid Clustering Algorithm Using Rough and Fuzzy Sets. Fundamenta Informaticae 80, 475–496 (2007)
27. Maji, P., Pal, S.K.: Rough Set Based Generalized Fuzzy C-Means Algorithm and Quantitative Indices. IEEE Transactions on System, Man and Cybernetics, Part B, Cybernetics 37, 1529–1540 (2007)
28. Dunn, J.C.: A Fuzzy Relative of the ISODATA Process and its Use in Detecting Compact, Well-Separated Clusters. Journal of Cybernetics 3, 32–57 (1974)
29. Krishnapuram, R., Keller, J.M.: A Possibilistic Approach to Clustering. IEEE Transactions on Fuzzy Systems 1, 98–110 (1993)
30. Krishnapuram, R., Keller, J.M.: The Possibilistic C-Means Algorithm: Insights and Recommendations. IEEE Transactions on Fuzzy Systems 4, 385–393 (1996)
31. Barni, M., Cappellini, V., Mecocci, A.: Comments on A Possibilistic Approach to Clustering. IEEE Transactions on Fuzzy Systems 4, 393–396 (1996)
32. Pal, N.R., Pal, K., Keller, J.M., Bezdek, J.C.: A Possibilistic Fuzzy C-Means Clustering Algorithm. IEEE Transactions on Fuzzy Systems 13, 517–530 (2005)
33. Haralick, R.M., Shanmugam, K., Dinstein, I.: Textural Features for Image Classification. IEEE Transactions on Systems, Man and Cybernetics SMC-3, 610–621 (1973)
34. Weszka, J.S., Rosenfeld, A.: Histogram Modification for Threshold Selection. IEEE Transactions on System, Man, and Cybernetics SMC-9, 62–66 (1979)
35. Rangayyan, R.M.: Biomedical Image Analysis. CRC Press, Boca Raton (2004)
36. Otsu, N.: A Threshold Selection Method from Gray Level Histogram. IEEE Transactions on System, Man, and Cybernetics 9, 62–66 (1979)
37. Banerjee, M., Mitra, S., Pal, S.K.: Rough Fuzzy MLP: Knowledge Encoding and Classification. IEEE Transactions on Neural Networks 9, 1203–1216 (1998)
38. Lingras, P., West, C.: Interval Set Clustering of Web Users with Rough K-means. Journal of Intelligent Information Systems 23, 5–16 (2004)
39. Bezdek, J.C., Pal, N.R.: Some New Indexes for Cluster Validity. IEEE Transactions on System, Man, and Cybernetics, Part B 28, 301–315 (1988)
40. Pal, S.K., Ghosh, A., Sankar, B.U.: Segmentation of Remotely Sensed Images with Fuzzy Thresholding, and Quantitative Evaluation. International Journal of Remote Sensing 21, 2269–2300 (2000)

Approximation Schemes in Logic and Artificial Intelligence

Victor W. Marek and Mirosław Truszczyński

Department of Computer Science
University of Kentucky
Lexington, KY 40506-0046, USA

Abstract. Approximate reasoning is used in a variety of reasoning tasks in logic-based artificial intelligence. In this paper we present several such reasoning schemes and show how they relate and differ from the approach of Pawlak's Rough Sets.

1 Introduction

Humans reason more often than not with incomplete information. The effect is that the conclusions must often be revised, and treated as approximate. Frequently we face the following situation: some features of objects of interest are positively established (based on observations and on domain properties), while other are known to be false. But there remains a "grey area" of features of objects of interest that are not determined by the current knowledge. In this paper we discuss several schemes that have been proposed in the literature for handling approximate reasoning when available knowledge may be incomplete. They include rough sets [21], approximation for propositional satisfiability [26], approximation semantics for logic programs including brave and skeptical answer-set semantics, Kripke-Kleene semantics and well-founded semantics [6,14,27], the semantics of repairs in databases [1], knowledge compilation of propositional theories [26], and least- and largest- pair of fixpoints for the operator associated with a Horn program [15]. For some of these, we will be able to show that they fit into the rough set paradigm.

2 Abstract Settings for Approximations

Let L be a lattice of elements (known or unknown to us) and let $X \subseteq L$. The problem we are interested in is that of describing X. The difficulty is that a language precise enough to represent every element in the lattice may not be available to us. Even if we have such a language, the descriptions may be inefficient (of large size or computationally unwieldy). Thus, we will represent X in terms of its approximations.

Let $X \subseteq L$. A pair $\langle a, b \rangle$ of elements of L is an *approximation* for X if for every $x \in X$, $a \leq x \leq b$. Approximations can be ordered by the *precision* (or *Kleene*) ordering [13] as follows:

$$\langle a_1, b_1 \rangle \preceq_{pr} \langle a_2, b_2 \rangle \text{ if } a_1 \leq a_2 \text{ and } b_2 \leq b_1.$$

If $a \leq b$, then there are non-empty subsets of L approximated by $\langle a, b \rangle$. If it is not the case that $a \leq b$, the only set approximated by $\langle a, b \rangle$ is the empty set. We call approximations of the first type *consistent* and all others — *inconsistent.* We denote the set of all approximations by L^2. We observe that approximations $\langle a, a \rangle$ are maximal consistent approximations (with respect to the ordering $\preceq_{pr}$). We also note that $\langle L^2, \preceq_{pr} \rangle$ is a lattice. If L is a complete lattice, L^2 is complete, too.

From now on we focus exclusively on complete lattices. Let L be a complete lattice. By $Sub(L)$ we denote the set of all sublattices K of L that have the same greatest and smallest elements as L. This set is ordered by inclusion (and in fact, it is a complete lattice, too).

By $\langle K^l(X), K^u(X) \rangle$ we denote the greatest approximation to X with respect to $\preceq_{pr}$ in K^2. It is clear that for every $X \subseteq L$, there is an approximation of X in K^2 and so the notion above is well defined. We have the following property. For every $K, M \in Sub(L)$, if $K \subseteq M$ then

$$\langle K^l(X), K^u(X) \rangle \preceq_{pr} \langle M^l(X), M^u(X) \rangle.$$

Descriptions of elements of K may be simpler than descriptions of elements of M (or available to us, while the other may be unavailable). Thus, it may be more practical to approximate X based on K rather than M. However, there is a trade-off. The precision of the approximation may (and in general, will) go down. We will show below that these concepts are abstract algebraic generalization of the concept of a rough set by Pawlak. We note that a closely related approach was studied in [9].

The setting we just described uses "coarsening" of a lattice to provide a practical way to approximate sets. There is also another general setting in which approximations of sets of lattice elements arise naturally. Namely, often sets of lattice elements are defined by means of lattice operators. In such case, the approximation theory [3] applies. Let us again consider a complete lattice L and an operator O on L. The goal is to describe (or at least approximate) the set of all fixpoints of O.

The key concept of the approach proposed in [3] is that of an approximation operator. An operator A on L^2 is an approximation operator for O if it is $\preceq_{pr}$-monotone, symmetric and if $A(x, x) = (O(x), O(x))$, for every $x \in L$. The fact that A is $\preceq_{pr}$-monotone implies that its least fixpoint exists and can be computed by iterating A starting with the $\preceq_{pr}$-least element of L^2, $\langle \bot, \top \rangle$. An important property of this least fixpoint is that it approximates all fixpoints of O.

In this setting, the quality of the approximation depends on the accuracy of the operator A. We say that an approximating operator A is at least as precise as an approximating operator B ($B \preceq_{pr} A$, in symbols) if for every $x, y \in L$, $B(x, y) \preceq_{pr} A(x, y)$. It is known that every operator has approximating operators, and that the set of all approximating operators for O has a greatest element, called the *ultimate* approximation operator [3]. Thus, in this setting, to approximate fixpoints of O we need to select an approximating operator for O, which is accurate enough and also for which it is easy to compute its least fixpoint. These two requirements may be hard to reconcile, giving rise to interesting trade-offs. Some of the specific examples discussed below are instantiations of this abstract schema.

We conclude this section by noting that one might combine the two approximation schemas. Namely, to approximate fixpoints of an operator O on L one might construct a simpler lattice K and an operator O' on K so that fixpoints of O' provide some information on fixpoints of O. For instance, K could be the quotient lattice for L with respect to a congruence relation r and O' could be defined by $O'([x]_r) = [O(x)]_r$. Then, clearly, if x is a fixpoint of O then $[x]_r$ is a fixpoint of O'. Consequently, approximating fixpoints of O' provides insights into fixpoints of O. To the best of our knowledge this schema has not been studied so far.

2.1 Rough Sets

Rough sets are special approximations. Let D be a finite set of objects (universe) and let $L(D)$ be the lattice of all subsets of D. In this section, *approximations* are pairs of elements from $L(D)$.

Every equivalence relation r on D determines a sublattice of $L(D)$ consisting of unions of cosets of r. We denote this sublattice $L(D, r)$. We note that $L(D) = L(D, =)$, where $=$ is the identity relation on D. We also note that all lattices $L(D, r)$ have the same least and greatest element ($\emptyset$ and D, respectively).

Now, for every $X \subseteq D$, *Pawlak's approximation* (or the *rough set* associated with X) is defined as an approximation $\langle X^l, X^u \rangle$ where: X^l is the union of all cosets of r contained in X, and X^u is the union of all cosets of r that have a nonempty intersection with X. It is characterized [19] as the $\preceq_{pr}$-largest approximation in $L(D, r)$ of X.

The collection of equivalence relations on D (not necessarily finite) determines a complete (non-distributive) lattice, with the refinement ordering $\sqsubseteq$. Specifically, $r_1 \sqsubseteq r_2$ if every coset of r_1 is the union of cosets of r_2. Let $r_1 \sqsubseteq r_2$ be two equivalence relations on D. Then $L(D, r_2)$ is a sublattice of $L(D, r_1)$. Our general results from the previous section imply that for every subset X of D, the Pawlak rough sets determined by r_1 and r_2, respectively, $\langle X_1^l, X_1^u \rangle$ and $\langle X_2^l, X_2^u \rangle$ are related as follows:

$$\langle X_1^l, X_1^u \rangle \preceq_{pr} \langle X_2^l, X_2^u \rangle.$$

In other words, the ordering $\sqsubseteq$ in the lattice of equivalence relations on D induces the ordering $\preceq_{pr}$ in the corresponding Pawlak approximations.

We note that an interesting possibility of establishing approximations of concepts in terms of rough sets is to learn them from negative and positive examples (cf. [23]). We also note that extensive discussion of lattice-theoretical notions as they relate to approximations in the context of rough sets can be found in [10].

2.2 Propositional Satisfiability

We consider a fixed set of propositional variables At. A valuation of At is any mapping of At into $\{0,1\}$. We can identify valuations with the subsets of At as follows. We identify a valuation v with the set $M \subseteq At$ so that $v = \chi_M$, i.e. $M = \{p : v(p) = 1\}$. We write v_M for the valuation v that determines M.

Obviously, each theory T represents the set of its models, that is, its satisfying valuations. However, the size of this set may be exponential in the size of T and so, it may be infeasible to represent it explicitly. However, T also determines a more concise and effective *approximation* to its set of models (by the "effective" approximation we mean that membership tests for the lower and upper parts are polynomial).

Namely, let T be a consistent set of formulas of the propositional language $\mathcal{L}_{At}$. Then T determines an approximation $\langle X_1, X_2\rangle$ as follows: $X_1 = \{p : T \vdash p\}$, and $X_2 = \{p : T \not\vdash \neg p\}$. This approximation has the property that $X_1 \subseteq M \subseteq X_2$ for every model M of T. Let us denote this "canonical" approximation of models of T by $\langle \underline{T}, \overline{T}\rangle$.

To represent the above construction of approximations in terms of rough sets, let us consider the following relation $\tilde{=}$ defined in the set of propositional variables At:

$$p \tilde{=} q \quad \text{if for all } M \models T,\ M \models p \equiv q$$

Then, clearly, $\tilde{=}$ is an equivalence relation in At. There are exactly three equivalence classes of $\tilde{=}$: $\{p : T \models p\}$, $\{p : T \models \neg p\}$, and the third class, that is the complement of the above two. The set $\underline{T}$ is the first class, the set $\overline{T}$ is the union of the first and the third class.

We have the following property of canonical approximations of theories T_1, T_2 that are consistent and such that $T_2 \models T_1$:

$$\langle \underline{T}_1, \overline{T}_1\rangle \preceq_{pr} \langle \underline{T}_2, \overline{T}_2\rangle.$$

In other words, not only the set of models of T_2 is a subset of the set of models of T_1. In addition, the canonical approximation of the theory T_2 is $\preceq_{pr}$-bigger than that of T_1. The maximal approximations (i.e. Pawlak's rough sets in this case) are the complete consistent theories.

Clearly, a class of consistent theories, say $\mathcal{T}$, determines the class of consistent approximations, say $A(\mathcal{T})$. By our comment above, if $\mathcal{T}_1$ and $\mathcal{T}_2$ are two classes of consistent theories and for every theory $T \in \mathcal{T}_1$ there is a theory $T' \in \mathcal{T}_2$ such that $T' \models T$, then the class $\mathcal{T}_2$ generates approximations at least as precise as the class $\mathcal{T}_1$.

As an illustration, we note that if $\mathcal{T}$ consists only of $\emptyset$, it generates only a trivial approximation $\langle \emptyset, At\rangle$. If $\mathcal{T}$ consists of definite Horn theories, then $\mathcal{T}$ generates approximations of the form $\langle M, At\rangle$, where $M \subseteq At$.

2.3 Approximating Finite Herbrand Structures

Reduction of truth in *finite* relational structures to propositional satisfiability yields a construction of *approximating pairs of relational structures over a finite domain.* We outline this construction briefly.

Let us fix a finite set of constants A and an algebra $\mathcal{A}$ with the universe A. All the relational structures under consideration will have $\mathcal{A}$ as its underlying algebra. To simplify presentation we will not write function symbols in our formulas at all. With this assumption, the atomic *sentences* of the predicate calculus language $\mathcal{L}$ are of the form

$$P(a_1, \ldots, a_s)$$

where s is the arity of the predicate symbol P. These atomic sentences of $\mathcal{L}$ can be treated as propositional variables. We will not distinguish between these two different entities. Let At be the set of these propositional variables, and let $\mathcal{L}'$ be the propositional language associated with At. We then define the following translation t of formulas of *predicate* language $\mathcal{L}$ to $\mathcal{L}'$:

1. $t(P(a_1, \ldots, a_s)) = P(a_1, \ldots, a_s)$
2. $t(\neg\Phi) = \neg t(\Phi)$
3. $t(\Phi \vee \Psi) = t(\Phi) \vee t(\Psi)$
4. $t(\forall_x \Phi) = \bigwedge_{a \in A} t(\Phi\binom{x}{a})$
5. $t(\exists_x \Phi) = \bigvee_{a \in A} t(\Phi\binom{x}{a})$

Here, $\Phi\binom{x}{a}$ is the result of substitution of constant a for every occurrence of variable x in Φ. Given a theory T in $\mathcal{L}$, $t(T) = \{t(\Phi) : \Phi \in T\}$. It is easy to see that the existence of a model (with $\mathcal{A}$ as underlying algebra) of a theory T is equivalent to the satisfiability of the propositional theory $t(T)$ and in fact, a structure $\mathcal{M}$ is a model of a theory T if and only if the set of variables $M_{\mathcal{M}}$ consisting of atomic sentences of $\mathcal{L}$ true in $\mathcal{M}$ satisfies $t(T)$. This fact allows to *pull-back* from the valuations for the language $\mathcal{L}'$ to relational structures with $\mathcal{A}$ as an underlying algebra.

Now, let us look at the two sets of propositional variables: $\underline{t(T)}$ and $\overline{t(T)}$. These two collection of atoms were introduced above in Section 2.2. These two sets of propositional variables determine, via pull-back, two relational structures $\mathcal{M}_1$ and $\mathcal{M}_2$. For every relational symbol P of $\mathcal{L}$ we have

$$P^{\mathcal{M}_1} \subseteq P^{\mathcal{M}_2}.$$

We write it as $\mathcal{M}_1 \subseteq \mathcal{M}_2$. Let us observe that we do not claim that any of the structures $\mathcal{M}_1$ or $\mathcal{M}_2$ is a model of T. The equivalence property stated above implies the following approximation property: for *every* structure $\mathcal{S}$ with the underlying algebra $\mathcal{A}$, such that $\mathcal{S} \models T$,

$$\mathcal{M}_1 \subseteq \mathcal{S} \subseteq \mathcal{M}_2.$$

Our discussion of the rough set concept related to the approximation in propositional satisfiability allows us (via pull-back) to find an equivalence relation in the set of all structures with the underlying algebra $\mathcal{A}$.

2.4 Knowledge Compilation

Many tasks in knowledge representation and reasoning reduce to the problem of deciding, given a propositional CNF theory T and a propositional clause φ, whether $T \models \varphi$. This task is coNP-complete. As a way to address this computational hurdle Kautz and Selman [26] proposed an approach in which T is compiled off-line, possibly in exponential time, into some other representation, under which the query answering would be efficient. While there is an initial expense of the compilation, if the query answering task is frequent that cost will eventually be recuperated.

In this scheme, an approximation to a theory T is a pair of theories (T', T'') such that

$$T' \models T \models T''$$

If (T', T'') is an approximation to T, then $T \models \varphi$ if $T'' \models \varphi$ and $T \not\models \varphi$ if $T' \not\models \varphi$. In other words,

$$\{\varphi\colon T'' \models \varphi\} \subseteq \{\varphi\colon T \models \varphi\} \subseteq \{\varphi\colon T' \models \varphi\}.$$

Desirable approximations are, on the one hand, "tight", that is, $\{\varphi\colon T' \models \varphi\} \setminus \{\varphi\colon T'' \models \varphi\}$ is small, and on the other hand, support efficient reasoning. Concerning the latter point, if U is a Horn theory and φ is a clause, then $U \models \varphi$ can be decided in polynomial time. Therefore, we define *approximations* to be pairs (T', T''), where T' and T'' are Horn theories such that $T' \models T''$.

A key problem is: given a CNF theory T, find the most precise Horn approximation to T. This problem has been studied in [26]. It turns out that there is a unique (up to logical equivalence) Horn least upper bound. However, there is no greatest Horn lower bound. The set of Horn lower approximations has, however, maximal elements.

2.5 Approximating Semantics for Logic Programs

Logic Programming studies semantics of *logic programs*, i.e. sets of *program clauses*. In the simplest case those are expressions of the form $p \leftarrow q_1, \ldots, q_m, \neg r_1, \ldots, \neg r_n$. The meaning of such clause is, informally, this: "if $q_1, \ldots, q_m$ have been derived, and none of $r_1, \ldots, r_n$ has, or ever will be, then derive p" (various different meanings are also associated with program clauses). It is currently commonly assumed that the correct semantics of a logic program (i.e. set of program clauses as above) is provided by means of fixpoints of Gelfond-Lifschitz operator, GL_P. Those fixpoints are called *stable models* of P [7], and more recently *answer sets* for P. The operator GL_P is antimonotone, thus existence of fixpoints of GL_P is not guaranteed. However the operator GL_P^2 is monotone and consequently, possesses a least and largest fixpoints.

A number of approximation schemes for stable semantics of logic programs has been proposed. A historically earliest proposal is the so-called Kripke-Kleene approximation ([14,6]). In this approach, one defines a *three-valued* van-Emden-Kowalski operator $\mathcal{T}_P$. That operator is monotone in the ordering $\preceq_{pr}$, and consequently possesses a least $\preceq_{pr}$ fixpoint. That fixpoint (which can be treated as an approximation) approximates all stable models of the logic program P. A stronger approximation scheme has been proposed in [27], and is called a (three-valued) *well-founded model* of the program. Essentially, that model is defined by means of the least and largest fixpoint of GL_P^2. Like the Kripke-Kleene fixpoint, the well-founded approximations provides an approximation to all stable models of the program. More recently, [4] introduced the *ultimate* approximation schema which, in general, is more precise that the well-founded one.

Of course, one can assign to a logic program P also the $\preceq_{pr}$-greatest (most precise) approximation that approximates *all* stable models. The lower bound in this approximation is the intersection of all stable models and the upper bound is given by their union. Let us denote by KK_P the Kripke-Kleene approximation, by WF_P the well founded approximation, by U_P the ultimate approximation, and by A_P the most precise approximations of all stable models. Then,

$$KK_P \preceq_{pr} WF_P \preceq_{pr} U_P \preceq_{pr} A_P$$

and examples can be given where all the above relationships are strict (we note in passing that A_P is consistent if and only if P has stable models).

The complexity of computing each of these approximations is also different, in general. The Kripke-Kleene approximation can be computed in linear time and the well-founded approximation in polynomial time (but it is not known whether linear-time algorithms exist). For a broad class of programs, the ultimate approximation can be computed in polynomial time. However, no polynomial time algorithms are known in general and the problem is coNP-hard. Finally, even for very simple classes of programs computing the most precise approximation is coNP-hard. These properties give rise, as in several places before, to interesting trade-offs between the precision of an approximation and the complexity of computing it.

We note the the Kripke-Kleene approximation KK_P approximates not only all stable models of P but also all supported models of P. In the case when P is a consistent Horn program (possibly with constraints), the fixpoint KK_P is given by the pair (S_l, S_u), where S_l is the least and S_u is the greatest supported model of P (which are guaranteed to exist in case of Horn program).

2.6 Approximating Possible-World Structures

The language of modal logic with the semantics of autoepistemic expansions and extensions [3] provides a way to describe approximations to possible-world structures. Let us consider a theory T in a language of propositional modal logic [8,24]. The theory T is meant to describe a possible-world structure providing the account of what is known and what is not known given T.

Since T may be incomplete, there may be several possible-world structures one could associate with T, that we will refer to as *intended* ones. Autoepistemic logic provides a specific definition of such structures; other nonmonotonic modal logics provide alternative notions [18]. To reason about the epistemic content of T by means of autoepistemic logic one has two possibilities: to compute all intended possible-world structures for T according to the semantics of the autoepistemic logic, or to compute an approximation to the epistemic content of T common to all these structures. The former is computationally complex, being a Σ_2^P-task. Hence, the latter is often the method of choice.

At least three different approximations can be associated with T, Kripke-Kleene approximation, the well-founded approximation and the ultimate approximation, listed here according to the precision, with which they approximate possible-world structures of T [3,4]. It is worth noting that the computational complexity of the first two of these approximations is lower that the complexity of computing even a (single) intended possible-world structure for T. To the best of our knowledge, the complexity of reasoning with ultimate approximations for autoepistemic logic has not been yet established.

2.7 Minimal Models Reasoning and Repairs in Databases

Approximations play an important role in the theory and practice of databases. In this paper, we regard a database as a finite structure of some language $\mathcal{L}$ of first-order logic that does not contain function symbols. Typically, legal databases are subject to *integrity constraints*, properties that at any time the database is supposed to have. In general, integrity constraints can be represented as arbitrary formulas of $\mathcal{L}$.

Databases are frequently modified over their lifetime. Updates create the possibility of entering erroneous data, especially that in most cases databases are modified by different users at different locations. Consequently, it does happen that databases do not satisfy the integrity constraints. Once such a situation occurs, the database needs to be *repaired* [1].

Let D be a database and let IC be a set of integrity constraints. A pair $R = (R^+, R^-)$ is a *repair* of D with respect to IC if $(D \cup R^+) \setminus R^- \models IC$ (the repair condition), and for every (Q^+, Q^-) such that $Q^+ \subseteq R^+$, $Q^- \subseteq R^-$, and $(D \cup Q^+) \setminus Q^- \models IC$, we have $Q^+ = R^+$ and $Q^- = R^-$ (the minimality condition). We write $R(D)$ for the database resulting from D by applying a repair R. We write $Rep(D, IC)$ to denote all repairs of D with respect to IC. The minimality condition implies that if (R^+, R^-) is a repair, then $R^+ \cap D = \emptyset$ and $R^- \subseteq D$.

Repairing a database D that violates its integrity constraints IC consists of computing a repair $R \in Rep(D, IC)$ and applying it to D, that is, computing $R(D)$. There are two problems, though. First, computing repairs is computationally complex (even in some simple settings deciding whether repairs exist is Σ_P^2-complete). Second, it often is the case that multiple repairs exist, which results in the need for some principled selection strategy.

These problems can be circumvented to some degree by modifying the semantics of the database. Namely, a database D with integrity constraints IC could be viewed as an *approximation* to an actual database D', not available explicitly but obtainable from D by means of a repair with respect to IC. The approximation to D' represented by (D, IC) is the pair of sets (D_l, D_u), where

$$D_l = \bigcap\{R(D) \colon R \in Rep(D, IC)\} \text{ and } D_l = \bigcup\{R(D) \colon R \in Rep(D, IC)\}$$

In other words, expressions (D, IC) define approximations and query answering algorithms have to be adjusted to provide best possible answers to queries to D' based on the knowledge of D_l and D_u only.

3 Further Work, and Conclusions

We discussed a number of approximation schemes as they appear in logic, logic programming, artificial intelligence, and databases. Doubtless there are other approaches to approximate reasoning that can be cast as approximations, and in particular rough sets. One wonders if there is a classification of approximations that allows to capture a common structure laying behind these, formally different, approaches. In other words, are there general classification principles for approximations? Are there categories of approximations that allow to classify approximations qualitatively?

Another fundamental issue is the use of languages that describe approximations. Pawlak [22] noticed that, in its most abstract form rough sets are associated with equivalence relations; each equivalence relation induces its own rough set notion. Such abstract approach leads to Universal Algebra considerations that have roots in [12] and has been actively pursued by Orłowska and collaborators [5,20,25]. One can find even more abstract versions within the Category Theory. But usually, the applications of rough sets and other approximation schemes can not choose its own language. For instance, more often than not (and this was the original motivation of Pawlak) the underlying equivalence relation is given to the application (for instance as the equivalence induced by an information system [16]). Then, and the literature of rough sets is full of such considerations, one searches for the coarser equivalence relations that are generated by various attribute reduction techniques. To make the point, these equivalence relations are not arbitrary, but determined by the choice of the language used for data description. This linguistic aspect of rough sets and approximations in general, needs more attention of rough set community.

Acknowledgments. This research was supported by the National Science Foundation under Grant IIS-0325063 and by the Kentucky Science and Engineering Foundation under Grant 1036-RDE-008.

References

1. Arenas, M., Bertossi, L.E., Chomicki, J.: Answer sets for consistent query answering in inconsistent databases. Theory and Practice of Logic Programming 3(4-5), 393–424 (2003)
2. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order. Cambridge University Press, Cambridge (1992)
3. Denecker, M., Marek, V., Truszczyński, M.: Uniform semantic treatment of default and autoepistemic logics. Artificial Intelligence Journal 143, 79–122 (2003)
4. Denecker, M., Marek, V., Truszczyński, M.: Ultimate approximation and its application in nonmonotonic knowledge representation systems. Information and Computation 192, 84–121 (2004)

5. Düntsch, I., Orłowska, E.: Beyond Modalities: Sufficiency and Mixed Algebras. In: [OS01], ch. 16 (2001)
6. Fitting, M.C.: A Kripke-Kleene semantics for logic programs. Journal of Logic Programming 2(4), 295–312 (1985)
7. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings. of the International Joint Conference and Symposium on Logic Programming, pp. 1070–1080. MIT Press, Cambridge (1988)
8. Hughes, G.E., Cresswell, M.J.: An Introduction to Modal Logic, Methuen (1968)
9. Iwiński, T.: Rough analysis in lattices. Working papers of the University of Carlos III, Madrid, Num., 91–23 (1991)
10. Järvinen, J.: Lattice Theory for Rough Sets. In: Peters, J.F., Skowron, A., Düntsch, I., Grzymała-Busse, J.W., Orłowska, E., Polkowski, L. (eds.) Transactions on Rough Sets VI. LNCS, vol. 4374, pp. 400–498. Springer, Heidelberg (2007)
11. Jonsson, B.: A Survey of Boolean Algebras with Operators. In: Algebras and Order, pp. 239–284. Kluwer, Dordrecht (1991)
12. Jonsson, B., Tarski, A.: Boolean Algebras with Operators. American Journal of Mathematics 73, 891–939 (1951)
13. Kleene, S.C.: Introduction to Metamathematics. North-Holland, Amsterdam (1967) (fifth reprint)
14. Kunen, K.: Negation in logic programming. Journal of Logic Programming 4(4), 289–308 (1987)
15. Lloyd, J.W.: Foundations of Logic Programming. Springer, Heidelberg (1987)
16. Marek, W., Pawlak, Z.: Information storage and retrieval systems, mathematical foundations. Theoretical Computer Science 1(4), 331–354 (1976)
17. Marek, W., Pawlak, Z.: Rough sets and information systems. Fundamenta Informaticae 7(1), 105–115 (1984)
18. Marek, V.W., Truszczyński, M.: Nonmonotonic Logic; Context-Dependent Reasoning. Springer, Berlin (1993)
19. Marek, V.W., Truszczyński, M.: Contributions to the Theory of Rough Sets. Fundamenta Informaticae 39(4), 389–409 (1999)
20. Orłowska, E., Szałas, A.: Relational Methods for Computer Science Applications. In: Selected Papers from 4th International Seminar on Relational Methods in Logic, Algebra and Computer Science (RelMiCS 1998); Studies in Fuzziness and Soft Computing, vol. 65. Physica-Verlag, Springer (2001)
21. Pawlak, Z.: Rough Sets. International Journal of Computer and Information Sciences 11, 341–356 (1982)
22. Pawlak, Z.: Rough Sets – theoretical aspects of reasoning about data. Kluwer, Dordrecht (1991)
23. Pawlak, Z., Skowron, A.: Rough sets: Some extensions. Information Sciences 177(1), 28–40
24. Rasiowa, H.: An Algebraic Introduction to Non-classical Logics. North-Holland, Amsterdam (1974)
25. SanJuan, E., Iturrioz, L.: Duality and informational representability of some information algebras. In: Polkowski, L., Skowron, A. (eds.) Rough Sets in Knowledge Discovery, Methodology and Applications, pp. 233–247. Physica-Verlag (1998)
26. Selman, B., Kautz, H.: Knowledge Compilation and Theory Approximation. Journal of the ACM 43(2), 193–224 (1996)
27. Van Gelder, A., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. Journal of the ACM 38(3), 620–650 (1991)

Decision Rule Based Data Models Using NetTRS System Overview

Marcin Michalak[1] and Marek Sikora[2]

[1] Silesian University of Technology, Institute of Computer Sciences,
44-100 Gliwice, Poland
Marcin.Michalak@polsl.pl
[2] Silesian University of Technology, Institute of Computer Sciences,
44-100 Gliwice, Poland
Marek.Sikora@polsl.pl

Abstract. The NetTRS system is a web service that makes induction, evaluation and postprocessing of decision rules possible. The TRS library is the kernel of the system. It allows to induce rules by means of the tolerance rough sets model. The NetTRS makes user interface of the TRS library available in the Internet. The main emphasis of the NetTRS system is placed on induction and postprocessing of decision rules. This article shows the architecture and the functionality of the system. This paper describes also the parameterization of algorithms that are implemented in the TRS library.

1 Introduction

Algorithms of rules induction based on examples are the part of the wider group of algorithms, that realize the learning by examples paradigm [8]. Recently, there could have been observed, the growth of machine learning methods applications in practice. Rules inductions algorithms are especially intensively used as a tool for the knowledge discovery from databases [5,7].

In relation to the observed trend, there arose a great number of programs that make it possible to perform rules induction or generate decision trees. The algorithms of rules induction and postprocessing are only the part of a wide range of analytical algorithms, implemented in the considered software. The solutions mentioned above can be divided into commercial ones, offered among the others by Salford-Systems, Statsoft, SAS Institute, SPSS, RuleQuest, and free available ones, developed at different academic centres — Weka [6] is the example of this kind of expanded system.

There is also a number of programs that perform rules induction using the rough set model (LERS [5], RSES [2], Rose [14], Rosetta [11], ARES [13] etc.). Almost all mentioned solutions offer, except analytical functions, more or less advanced tools for data managing (data import, data export, data cleaning, etc.) and results visualisation. It is worth to notice that rules induction algorithms are the standard part of the majority of database management systems (in particular solutions offered by the Oracle and Microsoft).

The NetTRS system differs from solutions mentioned previously due to the fact, that it is a web application which makes it possible to perform algorithms implemented in the TRS library [17]. The development of the library proceeded in the clearly defined direction: the main stress is laid on rules evaluation and postprocessing (generalization and filtration). The system presented in the paper does not contain functions that allow data managing. All data must be prepared in the specified format and divided into train, test and (in some cases) tune files.

The layout of the paper is following. Section 2 contains short description of the system architecture. Section 3 presents structure and content of scripts that control data analysis process. Section 4 shortly describes user interface. Finally, in Section 5 conclusions are included.

2 NetTRS System Architecture

The NetTRS system is a web application, implemented with the usage of the ASP.NET technology. The main purpose of this application is to make the computational potential of the TRS library available.

The TRS library has the possibility to control scripts interpreting, written as a sequence of commands. Each command, written in the script, is connected either with input/output procedure (ex. sending data for the analysis, results receiving) or with a special analytical algorithm realization.

The NetTRS system makes available graphical interface which creates control scripts for the TRS library in user friendly way. After script configuration and creation, the NetTRS system transmits it to executable form of the TRS library, the TRSExecutor program. Results of analysis are saved by the TRSExecutor

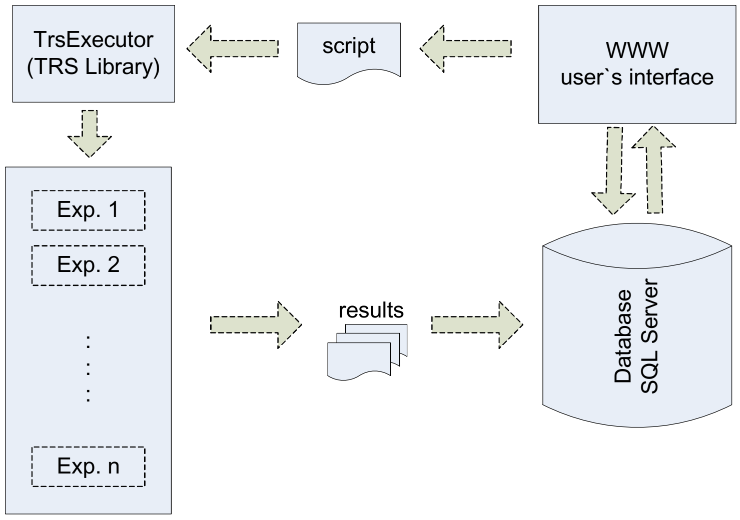


Fig. 1. The NetTRS system architecture

program in text output files, which can be made available to the user by the agency of the NetTRS system. Therefore, the TRS library is a computational kernel of the NetTRS system, the system itself is responsible for scripts creation, computations initializing and analysis results presentation. The NetTRS system architecture is presented in Fig. 1. The database shown in Fig. 1, stores the information about registered users, their experiments and calculation results.

When all analytical tasks are defined (with the usage of the web application) the control script is generated. The script and all other necessary data files are stored at the computational server. The separate program (TRSExecutor) is responsible for the cyclical database querying and starting the TRS library process for each new analysis task (experiment).

Single user can check status of experiments (queued/being executed/finished) via the website and download results of completed ones.

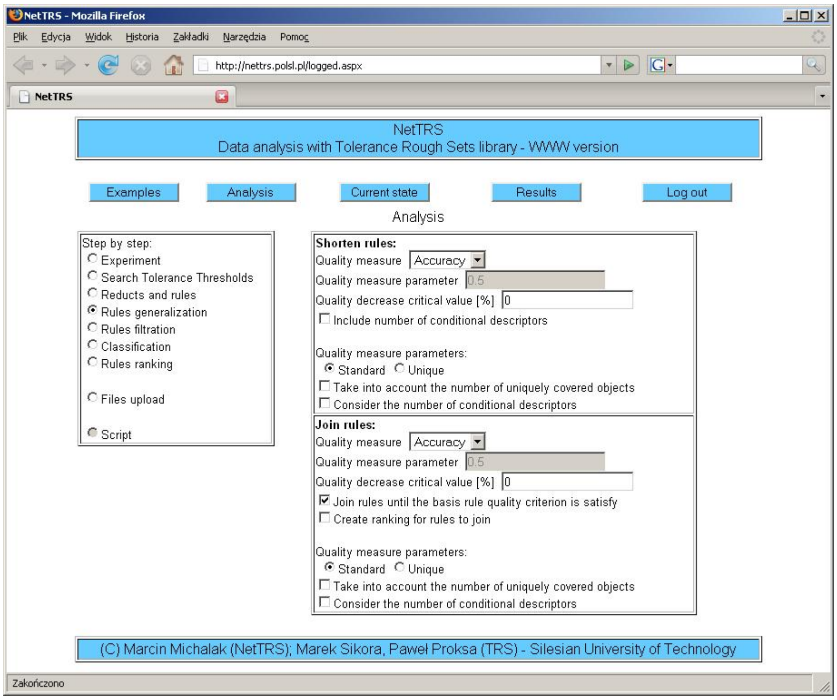
3 User Interface

The access to the NetTRS system is available only for users that have the personalized account (login and initial password), created by the system administrator. It is necessary to contact the service administrator to obtain the access to the system. After successful login, the user has the access to the subset of web sites (called "user panel") that helps him to use the system.

All web sites that compose the user available web application, can be divided into two parts: new analysis task defining and viewing and downloading results of the analysis.

The user panel has the analogical architecture as the tabbed one. The first tab *Analysis* allows to set the label of the new experiment, to upload all necessary data files and to configure selected algorithms. Names of consecutive steps of the analysis are placed on the left side of the web site presented in Fig. 2. Each step should be parameterized by the user, otherwise the step will be run with default values of parameters. Particular steps of the analysis contain following algorithms and functions:

– Experiment preparing – labelling the new experiment;
– Files upload — transferring data to the analysis;
– Tolerance thresholds searching — the genetic algorithm for tolerance thresholds searching, the heuristic algorithm for tolerance thresholds searching, loading thresholds from the file, saving thresholds to the file;
– Reducts and rules — determining all relative reducts, the heuristic algorithm finding a given number of relative reducts (including quasi-shortest ones [10]), calculating values of the attributes significance coefficient [12], finding all minimal decision rules (from objects-related relative reducts [24]), the heuristic algorithm that determines a given number of minimal decision rules (including rules from the quasi-shortest object-related relative reducts), the MODLEM algorithm [25], the modified version of the MODLEM algorithm [20], the RMatrix algorithm [22], loading the rules from the file, saving the rules to the file;


NetTRS - Mozilla Firefox

Plik Edycja Widok Historia Zakładki Narzędzia Pomoc

http://nettrs.polsl.pl/logged.aspx

NetTRS

NetTRS
Data analysis with Tolerance Rough Sets library - WWW version

Examples | Analysis | Current state | Results | Log out

Analysis

Step by step:
- Experiment
- Search Tolerance Thresholds
- Reducts and rules
- [✓ Rules generalization]
- Rules filtration
- Classification
- Rules ranking
- Files upload
- Script

Shorten rules:
Quality measure: Accuracy
Quality measure parameter: 0.5
Quality decrease critical value [%]: 0
☐ Include number of conditional descriptors

Quality measure parameters:
[✓ Standard] Unique
☐ Take into account the number of uniquely covered objects
☐ Consider the number of conditional descriptors

Join rules:
Quality measure: Accuracy
Quality measure parameter: 0.5
Quality decrease critical value [%]: 0
☑ Join rules until the basis rule quality criterion is satisfy
☐ Create ranking for rules to join

Quality measure parameters:
[✓ Standard] Unique
☐ Take into account the number of uniquely covered objects
☐ Consider the number of conditional descriptors

(C) Marcin Michalak (NetTRS); Marek Sikora, Paweł Proksa (TRS) - Silesian University of Technology

Zakończono

Fig. 2. User interface – setting parameters of the *Rules generalization* analysis step

– Rules generalization — algorithms of shortening and joining rules;
– Rules filtration — filtration algorithms: from coverage, forward, backwards, minimal quality;
– Classification — the voting classification algorithm;
– Rules ranking — creation of the file that contains the rules ranking, with respect to the rules quality measure selected by the user.

To realize the analysis task, it is necessary to send all needed data files (training, testing, and — if it is needed — tuning files) to the service. All files are text files, and the training file must contain a special header section. The format of all files is described in the tab *Examples*.

After all necessary files are successfully uploaded, the *Script* section becomes available. It shows the user the generated control script and makes its modification possible (for example, adding new command or changing one of the parameters). Selecting the button *Upload* makes the experiment realization start — putting in the experiment to the task queue. Since the user can define more than one experiment, the *Current state* tab makes user available to check the realization phase of each his initialised experiment.

After the single experiment and its all calculations are finished, all result files and data files are compressed to the archive file that can by downloaded from the tab *Results*. This page makes it also possible to see the content of the main result file by clicking the *View* button.

The primary result file, generated by the TRSExecutor, is the text file with the default name *out.txt*. Depending on algorithms selected during the experiment defining, the file contains: calculated tolerance thresholds, generated decision rules, decision rules after shortening, decision rules after joining, decision rules after filtration and the classification results.

The TRS library generates also some helpful partial result files, such as a decision rules with their quality measure values and the rules ranking (form the best to the worst) for all decision classes separately. As it was mentioned before, all files that are partial or final results of the experiment (including all data files), are packed to the archive file and made available to be downloaded by the user.

4 The Structure and Content of the Control Script

The control script, that is transmitted to the TRSExecutor program, is the most important control element of the analysis. It contains the sequence of commands which were selected and configured by agency of the NetTRS graphical interface. Each command starts with the algorithm name or the function name, and ends with the keyword *EndCommand*. The body of the command contains keywords reflecting names of parameters of particular algorithms and values of those parameters, for example:

```
LoadSystem from credit.train

CalcContRules
Entropy Yes
Quality Gain
QualityParam 0
EndCommand

Classify
Quality Coverage
QualityParam 0
File credit.test
EndCommand
```

Execution of commands sequence shown above produces following results:

- loading the training file named credit.train;
- rules induction by means of the MODLEM algorithm [25] with the following parameters: the conditional entropy as a conditional descriptors construction criterion; the *Gain* quality measure [15] as a criterion that decides when the process of adding new conditional descriptors to the conditional part of the decision rule should be stopped;

– making, by means of determined rules, classification of objects that are included in the credit.test file; the *Coverage* measure is used for solving classification conflicts.

Figure 3 shows the scheme of all possible analysis paths, available with the NetTRS system. The first step consists in uploading all data files to the system.

It is necessary to calculate or load the tolerance thresholds vector for relative reducts generating or decision rules generating (from relative reducts or with the usage of the RMatrix algorithm). The MODLEM algorithm does not need the thresholds vector. It is also possible to load decision rules directly from the file.

After the decision rules set is available, it is possible to apply algorithms of rules generalization and/or filtering. Each of these two algorithms can be performed more than once, in any sequence. It is also possible to omit this section and go to the classification step directly.

The current version of the NetTRS system GUI allows only the single usage of the each mentioned algorithm. It is a certain limitation in relation to the form

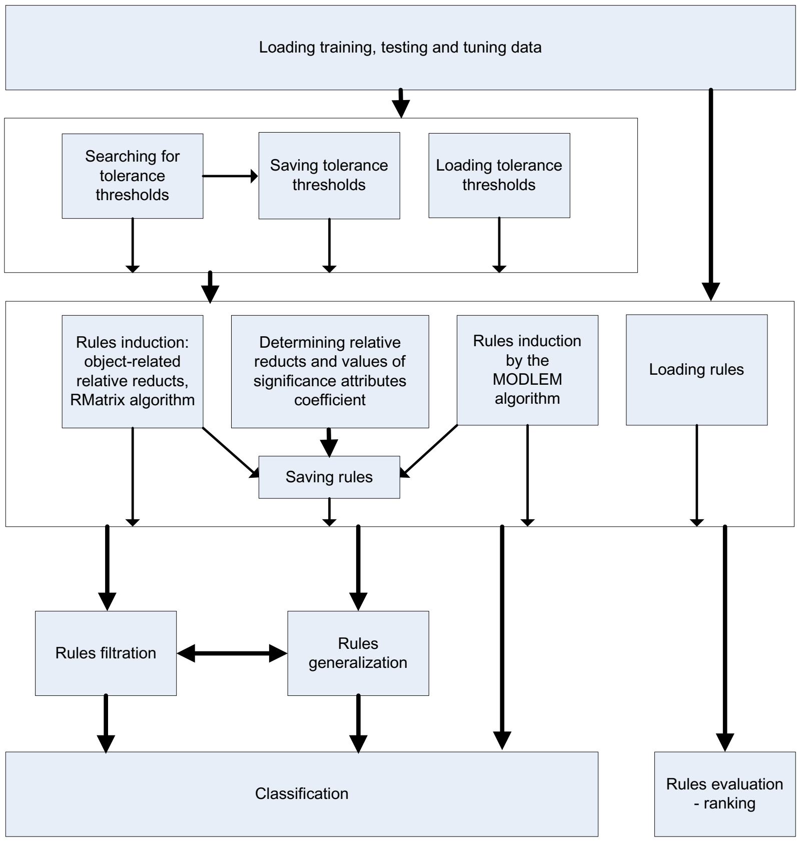

Loading training, testing and tuning data
Searching for tolerance thresholds
Saving tolerance thresholds
Loading tolerance thresholds
Rules induction: object-related relative reducts, RMatrix algorithm
Determining relative reducts and values of significance attributes coefficient
Rules induction by the MODLEM algorithm
Loading rules
Saving rules
Rules filtration
Rules generalization
Classification
Rules evaluation - ranking

Fig. 3. Possible analysis paths (data flow chart) using the NetTRS system

of scripts, that can be correctly interpreted by the TRS library. In generally, the number of each algorithm sections in the control script is unlimited (results are stored in the one output file *out.txt*). For example, if the control script contains two sections of rules generating, that precede two sections of rules shortening, than the set of rules generated by the second rules generating section is an input argument for the first section of shortening algorithm. Then the set of rules obtained as the result of first shortening rules section becomes an input argument for the second shortening rules section. In other words, the set of rules generated by the first generating rules section will be stored to the output files and will not be processed in the further analysis. The similar relation concerns other algorithms.

The exemplary control scripts, datasets, tolerance thresholds vectors, rules and sets that are results of analysis are available on the tab *Examples* in the NetTRS system.

4.1 The Parameterization of TRS Library Algorithms

All algorithms that are available via NetTRS system were described in the first part of this composed of two parts paper. The other publications, that also describe selected parts of TRS library algorithms, are [20,21,22,23]. This section contains the short overview of the algorithms parameters and their meaning.

Tolerance thresholds searching. Searching of the tolerance thresholds vector can be realized in two ways, using the genetic algorithm [9,27] or the heuristic one [23,26,27]. For each of them it is necessary to choose the distance measure (between values of attributes) and the function, that evaluates found vectors and decides about their quality (a specimen adaptation function).

The usage of the genetic algorithm requires the number of specimens in the population and probabilities of crossover and mutation. The minimal quality of specimens, sampled to the first population, can be also defined by the user, but this option significantly lengthens the duration of the first population sampling.

During the specimen adaptation function values computation in consecutive populations, it is possible to turn on the so–called scaling, that prevents from choosing specimens with the high value of adaptation function in the first phase of the algorithm running. The Boltzmann scaling [9] (1) was used in the implemented algorithm.

$$f'(s) = \frac{e^{\frac{f(s)}{T}}}{\frac{1}{p}\sum_{i=1}^{p} e^{\frac{f(i)}{T}}} \tag{1}$$

In equation (1), s is a specimen, f is an adaptation function, f' is a scaled adaptation function, p is a number of specimens in the population, T is a temperature value, that decreases in consecutive populations. The scaling needs from the NetTRS user to set the initial and final value of the temperature and define the decrease of the temperature in each step of the algorithm.

The genetic algorithm ends when all calculations for all defined by the user populations are finished or when the value of the adaptation function does not increase, in the defined by the user number of populations. In the second case there are two available parameters: the mean value of the specimens adaptation in the population and the adaptation value of the best specimen in the population.

If the heuristic algorithm that applies climbing strategy is selected, user should decide whether the vector with identical or different thresholds values for all attributes should be found and whether the number of steps is defined by the algorithm (all possible values will be considered that, for the set conditional attribute, change the size of the tolerance set of arbitrary training object) or by the user.

Relative reducts and decision rules. Before calculating relative reducts or minimal decision rules, it is necessary to define the number of awaited relative reducts and rules (in the case of rules induction — the number of minimal decision rules generated for each object). Fundamental options are *All*, which means calculating all relative reducts and minimal decision rules [12,24], and 1, which means calculating one quasi-shortest relative reduct or one quasi-shortest minimal decision rule for each object [10].

Considering the RMatrix algorithm [22], the rule quality measure [1,4] must be selected. In the case of the MODLEM algorithm [25] it is needed to specify, if the conditional entropy is a criterion that decides about the form of conditional descriptors and which rules quality measure is used as a stop criterion [20].

Turning the *From coverage* option on, which is available for algorithms of minimal decision rules generating and for the RMatrix algorithm, enables to determine the set of the decision rules, that is able to cover the training objects set. It is important to notice that this option functions in a way that the order of the rules generating is determined by the order of the objects in the training set. In other words, the rules induction does not start from the most representative objects from each decision class.

Rules generalization. The process of rules generalization can be realized by two different algorithms: the shortening and joining [21]. In each of them it is necessary to choose quality measure that controls the process of shortening or joining rules.

For the shortening algorithm user has to define, for each decision class separately, the maximal acceptable decrease of rules quality measure after shortening, in percentage (in particular, the value 0 means the lack of agreement for the quality decrease after shortening). In the case of the shortening algorithm it is also possible to determine, for each decision class separately, the minimal number of conditional descriptors in the decision rule. It causes that all rules, containing less descriptors than a mentioned value, will not be shortened.

In case of joining algorithm the user has to define the acceptable (in percentage) decrease of rules quality measure after joining, whether the set of joined rules should be joined later, and whether to create the special rules ranking before joining that determines the rules joining order. For the selected basis decision rule, the ranking of other decision rules from the same decision class is created. The top position in the mentioned ranking is for the rule (or rules) that covers the similar to the basis rule negative objects set and the different from the basis rule positive objects set.

Rules filtration. All available filtration algorithms use rules quality measures for evaluation of the filtered rules set. Therefore, the name of the applied rules quality measure must be defined for every filtration algorithm.

The *From coverage* filtration algorithm requires the information whether after adding the next rule to the filtered rules set, and after removing from the training set objects covered by the considered rule, the recalculation of the remaining rules qualities should be done.

For the *Minimal quality* algorithm, for each decision class separately, the minimal rules quality measure value should be defined. If the considered decision rule has the quality greater than the quality minimal value it remains in the filtered rules set.

The *Forward* filtration algorithm requires following parameters setting:

– The maximal decrease of the classification accuracy of the filtered rules set, defined for each decision class separately (in percentage); the reference point is the classification accuracy of the rules set before filtration;
– The maximal number of decision rules that create the description of each decision class;
– Choosing the criterion that decides about adding the rule to the filtered rules set; there are two possible criteria:
  - Add the rule to the filtered rules set if it causes the growth of the classification accuracy of the decision class which the rule describes and the growth of the general classification accuracy;
  - Add the rule r to the filtered rules set if the value of the expression (2) increases; in this expression RUL' means the filtered rules set together with the considered rule r, $accuracy(RUL', Tu)$ is the accuracy of the rules set RUL' obtained on the tuning objects set Tu, $\lambda \in [0, 1]$ and $simplicity(RUL') = (|RUL| - |RUL'|)/|RUL|$, where RUL means the unfiltered rules set.
– Parameters of the classification process of the tuning set that is used by the filtration algorithm.

$$evaluate(RUL') = \lambda\ accuracy(RUL', Tu) + (1 - \lambda)\ simplicity(RUL') \qquad (2)$$

The NetTRS system offers also the extended version of the *Forward* algorithm. The *Forward* algorithm makes it possible to store the information about the growth of classification accuracy in its succeeding steps in the result file.

The *Backwards* filtration algorithm requires, except for the parameters of the tuning set classification, the acceptable mean classification accuracy decrease of the filtered rules set. A value about which the difference between the most accurate and the least accurate decision class can increase is also the parameter of the Backwards algorithm.

Classification. The NetTRS system realizes the process of classification with the usage of the voting strategy. The classification may take place by exact matching rules or by nearest rules. User introduces the following classification parameters:

– the name of the rule confidence measure, that represents the strength of the rule vote during the classification,
– the type of the measure, used for calculating the distance between the rule and the testing object (Hamming, Euclidean),
– the maximal distance between the object and the decision rule (if the distance is less than the user defined than this rule participates in the voting),
– whether to divide "the number of votes" given for each decision class by the number of rules that participated in the voting (normalization).

Rules evaluation. The majority of mentioned methods and algorithms use rules quality measures. In the NetTRS system values of particular measures can be calculated in the standard way, with taking into consideration the number of objects covered uniquely by the rule and with taking into account the number of conditional descriptors occurring in the rule.

5 Conclusions and Further Works

In the paper we have presented technical details of the NetTRS system. It allows to induce and postprocess the decision rules obtained, among others, due to tolerance rough sets model. The NetTRS system is an extension of functionality of the TRS library that served, in its intension, an acceleration experiments executing during research works on rules generalization and rules filtration algorithms. The NetTRS system exploits a compiled version of the TRS library. Its functioning consists in automatic scripts generating (by user interface) for this library, and managing users of WWW service together with assigned calculation processes. The method of usege of the NetTRS system is described in the user‘s guide [18].

The present form the NetTRS system does not give access to all algorithms included in the TRS library. There are, among others, two interesting solutions: *ruleM5* which is a rule version of proposed by Quinlan the M5 [16] algorithm enabling to realize prediction tasks and successive modification of the MODLEM algorithm that adaptively selects (depend on actual characteristic of a data set) evaluation measures applied for rules creation [1,19]).

Further works on the system will focus on making it a more professional tool and will cover four areas:

– rewriting the TRS library from the C++ language on C# one and changing data structures connected with decision table storage (in order to accelerate work of the program and to omit a control script), and application methods of reducts calculation that does nor use discernability matrix [24];
– including the rule M5 and adaptative MODLEM algorithms in the service functionality;

– ability and access to experiments realization by means of train and test, and cross validation methodologies (a user will send only one data set);
– extending a functions of calculation tasks distribution on major number of computers.

In the context of the last task of the mentioned above, making parallel analytic algorithms is not less interesting than dispersions of calculations executed within a framework of one experiment. Here we can take pattern by an extension of the RSES program which is the program/library DIXER [3]. First works on this field have been already conducted by students of our faculty.

Both, the TRS library and the NetTRS system, still have experimental character only. At present the NetTRS system has a poor user interface — there is no possibility for data management (for example: data import from another than NetTRS format, data partition) — but it is one of the first systems, by means of which user can perform experiments (analyse data) with no need of analytical algorithms implementation.

To log into the server www.nettrs.polsl.pl/nettrs, it is necessary to have an account. It can be obtained after contact to service administrator Marcin Michalak[1].

References

1. An, A., Cercone, N.: Rule quality measures for rule induction systems– description and evaluation. Computational Intelligence 17, 409–424 (2001)
2. Bazan, J., Szczuka, M., Wróblewski, J.: A new version of rough set exploration system. In: Alpigini, J.J., Peters, J.F., Skowron, A., Zhong, N. (eds.) RSCTC 2002. LNCS (LNAI), vol. 2475, pp. 397–404. Springer, Heidelberg (2002)
3. Bazan, J., Latkowski, R., Szczuka, M.: DIXER– distributed executor for rough set exploration system. In: Ślęzak, D., Yao, J., Peters, J.F., Ziarko, W.P., Hu, X. (eds.) RSFDGrC 2005. LNCS (LNAI), vol. 3642, pp. 39–47. Springer, Heidelberg (2005)
4. Bruha, I.: Quality of Decision Rules: Definitions and Classification Schemes for Multiple Rules. In: Nakhaeizadeh, G., Taylor, C.C. (eds.) Machine Learning and Statistics, The Interface, pp. 107–131. Wiley, NY (1997)
5. Grzymała-Busse, J.W., Ziarko, W.: Data mining based on rough sets. In: Wang, J. (ed.) Data Mining Opportunities and Challenges, pp. 142–173. IGI Publishing, Hershey (2003)
6. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
7. Kubat, M., Bratko, I., Michalski, R.S.: Machine Learning and Data Mining: Methods and Applications. Wiley, NY (1998)
8. Michalski, R.S., Carbonell, J.G., Mitchel, T.M.: Machine Learning, vol. I. Morgan-Kaufman, Los Altos (1983)
9. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, USA (1996)
10. Nguyen, H.S., Nguyen, S.H.: Some Efficient Algorithms for Rough Set Methods. In: Proceedings of the Sixth International Conference, Information Processing and Management of Uncertainty in Knowledge-Based Systems, Granada, Spain, pp. 1451–1456 (1996)
11. Ohrn, A., Komorowski, J., Skowron, A., Synak, P.: The design and implementation of a knowledge discovery toolkit based on rough sets: The ROSETTA system. In: Polkowski, L., Skowron, A. (eds.) Rough Sets in Knowledge Discovery 1: Methodology and Applications, pp. 376–399. Physica-Verlag, Heidelberg (1998)

12. Pawlak, Z.: Rough Sets. Theoretical aspects of reasoning about data. Kluwer Academic Publishers, Dordrecht (1991)
13. Podraza, R., Walkiewicz, M., Dominik, A.: Credibility coefficients in ARES Rough Sets Exploration Systems. In: Ślęzak, D., Yao, J., Peters, J.F., Ziarko, W.P., Hu, X. (eds.) RSFDGrC 2005. LNCS, vol. 3642, pp. 29–38. Springer, Heidelberg (2005)
14. Prędki, B., Słowiński, R., Stefanowski, J., Susmaga, R.: ROSE – Software implementation of the rough set theory. In: Polkowski, L., Skowron, A. (eds.) RSCTC 1998. LNCS, vol. 1424, pp. 605–608. Springer, Heidelberg (1998)
15. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan-Kaufman, San Mateo (1993)
16. Quinlan, J.R.: Combining instance-based learning and model-based learning. In: Proceedings of the Tenth International Conference on Machine Learning (ML 1993), Massachusetts, Amherst, MA, USA, pp. 236–243 (1993)
17. Sikora, M.: Decision rules-based data models using NetTRS – methods and algorithms, NetTrs Website, Silesian Univesity of Technology, Gliwice, Poland (2006), `http://nettrs.polsl.pl/nettrs/Examples/NeTTRSalg.pdf`
18. Sikora, M., Michalak M.: Decision rules-based data models using NetTRS – users guide, NetTrs Website, Silesian Univesity of Technology, Gliwice, Poland (2007), `http://nettrs.polsl.pl/nettrs/Examples/Usersguide.pdf`
19. Sikora, M.: Adaptative application of quality measures in rules induction algorithms. In: Kozielski, S. (ed.) Databases, new technologies, vol. I. Transport and Communication Publishers (Wydawnictwa Komunikacji i Łączności), Warsaw, Poland (2007) (in Polish)
20. Sikora, M.: Rule quality measures in creation and reduction of data role models. In: Greco, S., Hata, Y., Hirano, S., Inuiguchi, M., Miyamoto, S., Nguyen, H.S., Słowiński, R. (eds.) RSCTC 2006. LNCS (LNAI), vol. 4259, pp. 716–725. Springer, Heidelberg (2006)
21. Sikora, M.: An algorithm for generalization of decision rules by joining. Foundation on Computing and Decision Sciences 30, 227–239 (2005)
22. Sikora, M.: Approximate decision rules induction algorithm using rough sets and rule-related quality measures. Theoretical and Applied Informatics 4, 3–16 (2004)
23. Sikora, M., Proksa, P.: Algorithms for generation and filtration of approximate decision rules, using rule-related quality measures. In: Proceedings of International Workshop on Rough Set Theory and Granular Computing (RSTGC 2001), Matsue, Shimane, Japan, pp. 93–98 (2001)
24. Skowron, A., Rauszer, C.: The Discernibility Matrices and Functions in Information systems. In: Słowiński, R. (ed.) Intelligent Decision Support. Handbook of applications and advances of the rough set theory, pp. 331–362. Kluwer Academic Publishers, Dordrecht (1992)
25. Stefanowski, J.: Rough set based rule induction techniques for classification problems. In: Proceedings of the 6th European Congress of Intelligent Techniques and Soft Computing, Achen, Germany, pp. 107–119 (1998)
26. Stepaniuk, J.: Optimizations of Rough Set Model. Fundamenta Informaticae 25, 1–19 (1998)
27. Stepaniuk, J.: Knowledge Discovery by Application of Rough Set Models. Institute of Compuer Sciences Polish Academy of Sciences, Reports 887, Warsaw, Poland (1999)

A Rough Set Based Approach for ECG Classification

Sucharita Mitra[1], M. Mitra[2], and B.B. Chaudhuri[1]

[1] Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata, India
{sucharita_r,bbc}@isical.ac.in
[2] Department of Applied Physics, Faculty of Technology, University of Calcutta, Kolkata, India
mmaphy@caluniv.ac.in

Abstract. An inference engine for classification of Electrocardiogram (ECG) signals is developed with the help of a rule based rough set decision system. For this purpose an automated data extraction system from ECG strips is being developed by using a few image processing techniques. A knowledge base is developed after consulting different medical books as well as feedback of reputed cardiologists on interpretation and selection of essential time-plane features of ECG signal. An algorithm for extraction of different time domain features is also developed with the help of differentiation techniques and syntactic approaches. Finally, a rule-based rough set decision system is generated using these time-plane features for the development of an inference engine for disease classification. Two sets of rules are generated for this purpose. The first set is for general separation between normal and diseased subjects. The second set of rules is used for classifications between different diseases.

Keywords: Rough set, knowledgebase, decision system, Electrocardiogram (ECG).

1 Introduction

In 1889 Waller first developed a method of recording [21, 62] the ECG voltage by capillary electrometer introduced by Lippman in 1875. The method was improved by using the string galvanometer discovered by Einthoven in 1903. In 1906 Einthoven [16] added a new dimension by introducing the concept of vectors to represent the ECG voltages. He also standardized the electrode locations for collecting ECG signals as right arm (RA), left arm (LA) and left leg (LL), and these locations are known after him as the standard leads of Einthoven or limb leads(see fig. 1). String galvanometer was replaced by electronic amplifiers around 1920s, which allowed the use of less sensitive and more rugged recording devices. Next, direct writing recorders, which used ink or pigment from a ribbon to record the ECG trace on a moving paper strip, were intrduced around 1946. Later on, a special heat sensitive paper was developed, which is now used almost

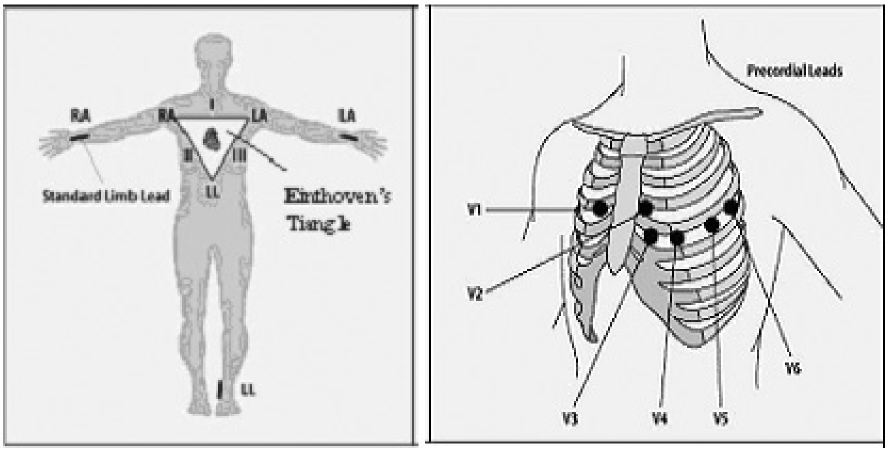


Fig. 1. The placement of the bipolar leads (Left image) and the exploratory electrode for the unipolar chest leads (Right image) in an electrocardiogram (ECG); (RA = right arm, LA = Left arm, LL = left leg)

exclusively as a recording medium for electrocardiograms. Modern direct writing electrocardiographs have a frequency range extending to over 0 - 100 Hz, which is quite adequate for clinical ECG recordings.

Wilson [70] in 1934, and, later, Frank [17] in 1955, have made considerable progress in the dipole theory of the heart. This is the first significant step to solve ECG inetrpretation problem analytically. The human body is assumed to be a uniform, homogeneous, isotropic conducting medium having the shape of a sphere containing a centric current dipole, which simulates the electrical activity of the heart. In addition, Wilson suggested that the three Einthoven leads be connected by means of three equal external resistors to an external node is now called Wilson Central Terminal (WCT). Using centric dipole model [55] WCT is shown to be at electrical zero of the system. He also introduced the unipolar limb leads (AV_R, AV_L, AV_F) and chest leads (V_1,V_2,... V_6). These unipolar leads along with Einthoven's bipolar limb leads are used in the modern 12 lead electrocardiogram (see fig. 1).

In the late 1960s, low cost microprocessors became available due to advent of integrated-circuit technology and work started to develop, which has immensely advanced during the last 40 years. As a result, the computerized ECG analysis began to emerge on experimental basis with the help of lower cost, on-line, real time computer systems since such systems became both economically and technically feasible for clinical use [13]. However, the research is still continue to boost the sophistication of the methodology and to condense the dimension of the hardware so that it becomes more mobile, accurate and helpful for both doctors and patients.

1.1 Raw ECG Data Extraction

The first step of computer analysis is to acquire digitized ECG data. For this purpose, some small processing algorithms are developed which can transfer continuous data recorded on paper to a digital time data, corresponding to those obtained by A/D converter [10, 68, 69]. An instrumentation scheme using Computer Aided Design and Drafting (Auto CAD) application package is developed by our group to capture ECG database. Here, a single channel strip chart recorder and a digitizer with tablet attached to the RS 422/432 port of the computer is used as input device. A digital plotter/printer is used as the output device [23]. Alternative system is reported in [5], and several others may be found in the literature.

1.2 Work on ECG Wave Segment Detection and Feature Extraction

ECG conveys information regarding the electrical function of the heart, by altering the shape of its constituent wave sesments, namely the P, QRS, and T waves (see fig. 2). Pattern recognition approaches are widely used for the detection and analysis of these waves. Direct signal pattern analysis, non-linear signal transformations, principal component analysis, and neural networks (NN) based techniques are used for ECG pattern recognition and classification [8, 15, 39, 67]. The *significant point* extraction algorithm, based on the analysis of curvature, is an example of direct signal analysis, that helps in both data reduction as well as pattern matching [11, 36]. Syntactic approaches are also used for ECG waveform analysis [19, 32]. In another study, a learning system is reported to grammatically classify biomedical patterns [20]. Other approaches for P and St segment detection are reported in [35, 61, 65].

In recent years, Wavelet transform has been used to decompose ECG signal for detection of P wave by neural network [72]. In another case, wavelet is employed to obtain a multiresolution representation of some example patterns for ECG signal structure extraction. Similarly, Neural Networks are trained with the wavelet-transformed templates providing an efficient detector even for temporally varying patterns within the complete time series [66]. Recently, multi-resolution wavelet transform, wavelet decomposition and continuous wavelet transform have been combined for ECG feature extraction [29, 33, 38, 41, 42].

QRS detection is an important task in time-plane ECG analysis, that helps in detection of other time-plane features more accurately. Hidden Markov Model [12], Wavelet transform [30] and Artificial Neural Network (ANN) [74] were used for the detection of QRS complexes from ECG signals. A slope vector waveform based QRS detection algorithm which is ideal for embedded real-time ECG monitoring is also reported [73]. Various QRS detectors are compared in [18].

Baseline (see fig. 2) detection is another essential task in ECG analysis, which aids for extraction of different time domain features. Most methods for baseline or isoelectric line detection are based on the assumption that the isoelectric level of the signal lies on the area ∼80 ms left of the R-peak, where the first derivative remains zero for at least 10 ms or minimum in the 20 ms segment [39, 40].

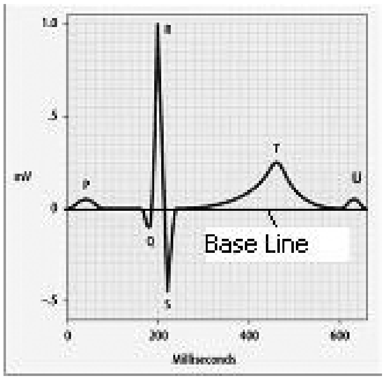


Fig. 2. A typical cycle of ECG signal

1.3 Studies on ECG Classification and Abnormality Detection

Considerable research has been done to assist cardiologists with their task of diagnosing the ECG recordings. The research field range from abnormality detections to fully automated ECG diagnosing systems. A wide range of techniques has been used, including statistical pattern recognition, Expert Systems, Artificial Neural Networks, Wavelet Transform, Fuzzy and Neuro-fuzzy Systems.

The computer task for ECG interpretation comprises of two distinct and sequential phases: feature extraction and classification. A set of signal measurements containing information for the characterization of the waveform is obtained by feature extraction methods. These waveform descriptors are then used to allocate the ECG to one or more diagnostic classes in the classification phase. These classifiers may be heuristic and use rules-of-thumb or employ fuzzy logic as a reasoning tool [14]. A classifier may also be statistical with the use of complex and even abstract signal feature probability to define discriminant functions for class allocation.

Many approaches have been proposed to generate Expert Systems for ECG diagnosis [1, 71]. An approach to intelligent ischaemia event detection is proposed based on ECG ST-T segment analysis. ST-T trends have been processed by a Bayesian forecasting approach using multistate Kalman filter [7]. Several Ischemia detection methods are proposed in [37, 40, 56]. Hermite functions and Self Organizing Maps(SOM) are used for clustering ECG complexes [34]. Another system for automatic analyzing of ECG is proposed in [71]. Expert's knowledgebase dependent Ischemia detection and automatic ECG interpretation techniques are described in [46, 58, 63].

More recently, artificial neural network techniques have been employed for signal classification [6]. Learning algorithms for two phase and three phase radial basis function (RBF) networks are proposed in [3]. Bi-group Neural Network classifiers are also utilized to examine independent feature vectors of ECG recordings for each diagnostic class and the outputs from the classifiers are fused together to produce single result [44].

A method is developed with wavelet transforms as features extractor and Radial Basis Function Neural Network (RBFNN) as classifier for arrhythmia detection [2]. Also, Fuzzy Adaptive Resonance Theory MAP (ARTMAP) is used to classify cardiac arrhythmias [25]. A hybrid neuro-fuzzy system for ECG classification of myocardial infarction is reported in [9].

For the past few years, rough set theory and granular computation has proved to be another soft computing tool which, in various synergetic combination with fuzzy logic, artificial neural network and genetic algorithms provide a stronger frame work to achieve tractability, low cost solution, robustness and close resembles with human like decision making. For example, rough-fuzzy integration is the basis of the Computational Theory of Perceptions (CPT), recently explained by Zadeh, where perceptions are considered to have fuzzy boundaries and granular attribute values. Similarly to describe different concept or classes, crude domain knowledge in the form of rules are extracted with the help of rough neural synergistic integration and encoded them as network parameters. Thus the initial knowledge base network for efficient learning has been built. In the case of granular computation every operation are done on granules (clump of similar objects or points), rather than on the individual data points. As a result, the computation time is greatly reduced. As the methodology is getting matured, several interesting applications of the theory have surfaced, also in medicine. For example, in a medical setting, sets of interest to approximate could be the set of patients with a certain disease or outcome, or the set of patients responsive to a certain treatment. Pawlak [48] used rough set theory in Bayes' theorem and showed that it can apply for generating rule base to identify the presence or absence of disease. Discrete Wavelet Transform and rough set theory were combined for classification of arrhythmia [31].

2 Basics of Electrocardiogram (ECG)

A pair of surface electrodes placed two different locations on the heart of the body will record a repeating pattern of changes in electrical "action potential" of the heart. The heart has four chambers namely left atrium, left ventricle and right atrium, right ventricle. As action potentials spread from the atria to the ventricles, the voltage measured between these two electrodes will vary in a way that provides a "picture" of the electrical activity of the heart. The nature of this picture can be varied by changing the position of the recording electrodes; different positions provide different perspectives, enabling an observer to gain a more complete picture of the electrical events. The body is a good conductor of electricity because tissue fluids contain a high concentration of ions that move (creating a current) in response to potential differences. Potential differences generated by the heart are thus conducted to the body surface where they are recorded by surface electrodes placed on the skin. The recording is called an electrocardiogram (ECG or EKG).

There are two types of ECG recording *leads.* The bipolar limb leads record the voltage between electrodes placed on the wrists and legs. These bipolar leads include lead I (right arm to left arm), lead II (right arm to left leg), and lead III (left arm to left leg). In the unipolar leads, voltage is recorded between a single *exploratory electrode* placed on the body and an electrode that is built into the electrocardiograph and maintained at zero potential (ground).

The unipolar limb leads are placed on the right arm, left arm, and left leg; these are abbreviated AV_R, AV_L, and AV_F, respectively. The unipolar chest leads are labeled one through six, starting from the midline position (fig. 1). There are thus a total of twelve standard ECG leads that "view" the changing pattern of the heart's electrical activity from different perspectives. This is important because certain abnormalities are best seen with particular leads and may not be visible with other leads.

As shown in fig. 2, each cardiac cycle produces three distinct ECG wave segments designated P, QRS, and T. When a heart muscle cell is stimulated, it begins to depolarize and the spread of depolarization through the atria causes a potential difference that is indicated by an upward deflection of the ECG line. When nearly half the mass of the atria is depolarized, this upward deflection reaches a maximum value, because the potential difference between the depolarized and unstimulated portions of the atria is at a maximum. When the entire mass of the atria is depolarized, the ECG returns to baseline, because all regions of the atria have the same polarity. In this way, the spread of atrial depolarization creates the P wave.

Similarly, conduction of the impulse into the ventricles creates a potential difference that results in a sharp upward deflection of the ECG line, which then returns to the baseline as the entire mass of the ventricles becomes depolarized. The spread of the depolarization into the ventricles is represented by the QRS wave. During this time the atria does repolarize i.e, it retuns to its resting state, but this event is hidden by the greater depolarization occurring in the ventricles. Finally, repolarization of the ventricles produces the T wave (fig. 2). The T wave may be notched or inverted in shape also. Sometimes another rounded wave, the U wave, follow the T wave. The exact significance of this wave is not clearly known. Functionally it represents the last phase of ventricular repolarization. Normally the prominent direction of U wave is the same as that of T wave. Negative U waves sometimes appear with positive T waves. This abnormal situation has been noted in left ventricular hypertrophy and myocardial ischemia.

3 Ischemic Heart Disease (IHD)

Different heart diseases that can be interpreted by ECG may be broadly classified into 4 major classes. They are: (a) Atrial and Ventricular Enlargement or Chamber Enlargement, (b) Ventricular Conduction Disturbance, (c) Ischemic Heart Disease (IHD) and (d) Cardiac Rhythm Disturbance. But statistical surveys indicate that IHD is a major health burden in India and other developing countries.

In this paper we concentrate on analysis and classification of IHD. Fig. 3 shows a cross section through the heart muscle, called cardium, having several layers. The innermost layer, called endocardium, is a layer of smooth lining cells.

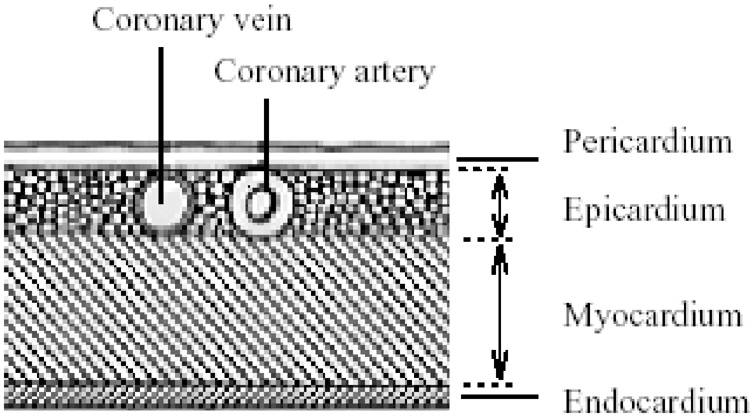


Fig. 3. Layers of the Heart Muscle

The myocardium is the mass of the heart muscle cells whose coordinated contraction causes the chambers of the heart to contract and pump blood. The next layer myocardium is thin in the atria, thicker in the right ventricle and thickest in the left ventricle. The epicardium is a fatty layer on the outer surface on the myocardium. The major coronary blood vessels, the vessels that supply blood to the heart itself, run through the epicardium. The outermost layer is the pericardium, actually two layers with a small amount of lubricating fluid between them, forming the pericardial sac which encloses the entire heart.

Myocardial cells require oxygen and other nutrients supplied by the coronary arteries. Severe narrowing or complete blockage of a coronary artery cause, the blood flow to be inadequate, then ischemia of the heart muscle develops. If the ischemia is more severe, permanent damage or necrosis (cell death) of a portion of heart muscle may occur. Myocardial Infarction (MI) refers to myocardial necrosis ("heart attack") which is usually caused by severe ischemia. Myocardial Ischemia or Infraction may affect the entire thickness of the ventricular muscle (transmural injury) or may be localized to the inner layer of the ventricle (subendocardial ischemia or infarction).

Transmural MI often(but not always) produces a typical sequence of ST-T changes and abnormal Q waves (duration is 0.04sec. or more in lead I, all three inferior leads[II, III, aV_F], or leads V_3 to V_6). The ST-T changes can be divided into two phases:

The acute phase of transmural MI is marked by ST segments elevations and sometimes tall positive T waves (hyper-acute T waves).

The evolving phase is characterized by the appearance of deeply inverted T waves in leads that show the hyperacute T waves and ST elevations.

3.1 Classification of MI

Infarction of the heart generally occur in left ventricle which is cone shaped and divided into 4 regions (Basal, Mid, Apical and Apex) of 17 segments (6 basal,

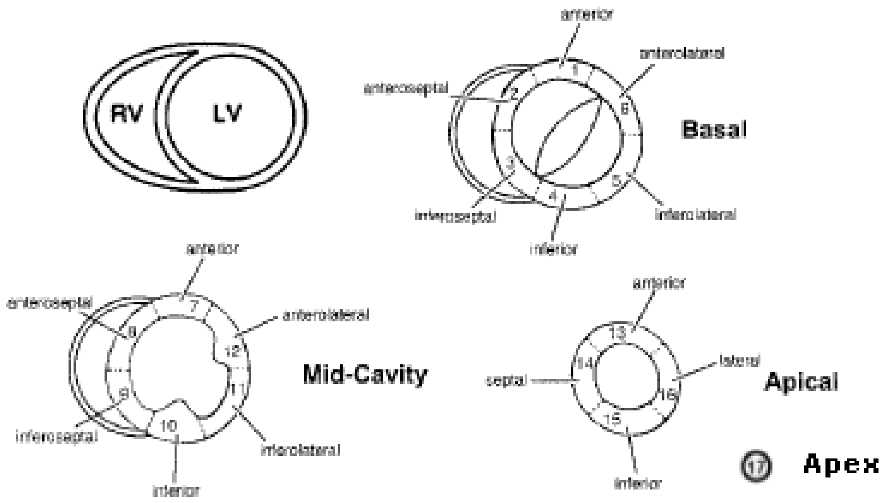


Fig. 4. 17 standard segments and 4 walls of left ventricular cone

6 medial, 4 apical and the apex) and 4 walls (Anterior, Inferior, Septal and Lateral)[fig. 4]. MI can be classified according to the location of the damage in the walls of the left ventricular cone. Hence, if the infarction or necrosis occur in the inferior wall then it will be classified as inferior wall infarction and typical infarction pattern is reflected in lead II, III and AV_F. Similarly, damage in anterior wall will be termed as anterior wall infarction and signal in standard lead I, AVL and all the precordial leads (V_1 to V_6) show the infarction pattern.

Beside these, there are two other walls known as septal (lead V_1, V_2) and lateral (lead I, AV_L,V_5, V_6). Damage in septal and lareral surfaces along with anterior or inferior surfaces may be termed as antero-lateral, antero-septal, infero-lateral, infero-septal MI and signal from all the leads oriented to these surfaces will show the typical infarction pattern.

4 Rough Sets

The theory of rough sets, introduced by Pawlak [49, 50] in 1982, has lately emerged as a key mathematical tool for managing ambiguity that arises from vague, noisy or partial information. It is methodologically significant to the domains of artificial intelligence and cognitive sciences, especially in the representation of reasoning with vague or imprecise knowledge, data classification, rule creation, machine learning, data mining, and knowledge discovery. The theory is also showing to be of substantial consequence in many other areas of applications [52, 53, 54].

4.1 Mathematical Basics of Rough-Set Theory

Rough set theory proposed by Pawlak [51], deals with imprecise or vague concepts. Central to the theory is an information system that can be viewed as a data table, whose columns are labeled by attributes, rows are labeled by objects of interest and entries of the table are the attribute values.

If, U and A are finite, nonempty sets where, U is the universe of objects and A is the set of attributes then, $S = (U,A)$ is an information table where every attribute $a \in A$ is associated with a set V_a, of its values, called the domain of a.

Any subset B of A will establish a binary relation $I(B)$ on U, called an *indiscernibility relation*, by satisfying the following conditions.

$x\ I\ (B)\ y$ if and only if $a\ (x) = a\ (y)$ for every $a \in B$, where $a(x)$ denotes the value of attribute a for object x. Obviously $I(B)$ is an equivalence relation.

The family of all equivalence classes of $I(B)$, i.e., a partition determined by B, will be denoted by $U/I(B)$, or simply by U/B; an equivalence class of $I(B)$, i.e., block of the partition U/B, containing x will be denoted by $B(x)$. If $(x, y) \in I\ (B)$ it is said that x and y are B-indiscernible (indiscernible with respect to B). Equivalence classes of the relation $I(B)$(or blocks of the partition U/B) are referred to as B-elementary sets or B-granules.

In rough set based methods, these granules are the basic building blocks about our knowledge of realty. The union of B-granules are known as B-definable sets.

Now, consider Xa proper subset of universe U. Two sets $B_*(X)$ and $B*(X)$, called the B-lower and the B-upper approximation of X, respectively, can be defined as

$$B_*(X) = \bigcup_{x \in U} \{B(x) : B(x) \subseteq X\}. \tag{1}$$

$$B^*(X) = \bigcup_{x \in U} \{B(x) : B(x) \cap X \neq \phi\}. \tag{2}$$

It is clear that B-lower approximation of a set is the union of all B-granules that are included in the set, whereas B-upper approximation of a set is the union of all B-granules that have a nonempty intersection with the set. The set

$$BN_B\ (X) = B^*\ (X) - B_*(X)$$

Is defined as the B-boundary region of X. If the boundary region of X is the empty set, i.e., $BN_B\ (X) = \emptyset$, then X is crisp (exact) set with respect to B. On the other hand, if $BN_B\ (X) \neq \emptyset$, X is referred to as rough (inexact) set with respect to B.

4.2 Rough-Set Description

In various rule based practical applications, rough set theory is used for getting the optimal number of appropriate rules needed for developing a classifier. From every information system, a subset of minimal attributes is generated which is known as *reduct*. Determination of reduct is a computationally expensive task. Different algorithms are available to generate rules from this reduct.

To describe such a system more precisely, consider a decision table expressed as S = (U,C, D) , where A i.e, the set of attributes are partitioned into two classes C, D $\subseteq$ A, called *condition* and *decision* attributes respectively. Every x

$\in$ U determines a sequence c_1 (x) , . . . , c_n (x), d_1 (x) , . . . , d_m (x) where {c_1, . . . , c_n} = C (conditions) and {d_1, . . . , d_m} = D (decisions).

The sequence will be called a decision rule induced by x (in S) and denoted by c_1 (x) , . . . , c_n (x) $\rightarrow$ d_1 (x) , . . . , d_m (x) . In short, C$\rightarrow_x$ D.

Thus, the decision table determines decisions, which must be taken, when some conditions are satisfied. In other words, each row of the decision table specifies a decision rule which determines decisions in terms of conditions.

The term supp$_x$ (C,D) = |C (x) $\cap$ D(x) | is called a *support* of the decision rule

C $\rightarrow_x$ D and the number σ_x (C,D) = $\frac{\sup p_x(C,D)}{|U|}$, will be referred to as the *strength* of the decision rule C $\rightarrow_x$ D.

Every decision rule C $\rightarrow_x$ D is allied with the *certainty factor* of the decision rule, denoted by cer$_x$ (C,D) and defined as

$$cer_x(C,D) = \frac{|C(x) \cap D(x)|}{|C(x)|} = \frac{\sup p_x(C,D)}{|C(x)|} = \frac{\sigma_x(C,D)}{\pi(C(x))}. \tag{3}$$

$$\text{where } \pi(C(x)) = \frac{|C(x)|}{|U|}.$$

The certainty factor may be interpreted as a conditional probability that y belongs to D(x) given y belongs to C (x), symbolically π_x (D|C) where y must be an object of the universal set.

If cer$_x$ (C,D) = 1, then C $\rightarrow_x$ D will be called a certain decision rule in S;

if 0 < cer$_x$ (C,D) < 1 the decision rule will be referred to as an uncertain decision rule in S.

Besides, a *coverage factor* of the decision rule is also used and denoted as cov$_x$ (C,D), defined as

$$cov_x(C,D) = \frac{|C(x) \cap D(x)|}{|D(x)|} = \frac{\sup p_x(C,D)}{|D(x)|} = \frac{\sigma_x(C,D)}{\pi(D(x))}. \tag{4}$$

$$\text{where } \pi(D(x)) = \frac{|D(x)|}{|U|}.$$

Similarly cov$_x$ (C,D) = π_x (C|D) .

If C $\rightarrow_x$ D is a decision rule then D $\rightarrow_x$ C will be called an inverse decision rule. The inverse decision rules can be employed to provide explanations (reasons) for decisions. Decision rules are normally represented in a form of “if ... then ...” implications. So decision table can be altered in a set of “if ... then ...” rules, called a decision algorithm. Using this decision algorithm, the optimal rules are generated which are used for development of the rule based classifier.

Another important factor in data analysis is to find out the *degree of dependency* γ (C, D) between condition attributes C and decision attributes D. It can be shown that D depends on C in a degree k (0 $\leq$ k $\leq$ 1) where C $\rightarrow_k$ D, if

$$k = \gamma(C,D) = \frac{|POS_C(D)|}{|U|}. \tag{5}$$

where, $\mathrm{POS}_C\ (\mathrm{D}) = \bigcup_{X \in U/D} C_*(X)$ is known as a positive region of the partition U/D with respect to C. Positive region is actually the set of all elements of U that can be individually classified to blocks of the partition U/D by means of C. If k=0, then D is independent on C. On the other hand, if k=1, D is fully dependent on C. Values 0< k < 1 denote partial dependency.

5 Materials and Methods of Analysis

The block diagram of the developed system is given in fig. 5. The detail methodologies are described below in step by step.

5.1 Development of ECG Data Extraction System

A software is being developed for getting pixel to pixel co-ordinate information of every ECG images with the help of few image processing techniques [43]. For development of off-line data extraction system [GUI based], the paper records are scanned by flat-bed scanner (HP Scanjet 2300C) to form image database in TIFF format. These TIFF formatted gray tone images are converted into two tone binary images with the help of a global thresholding technique on gray value histogram [22]. This method almost removes the background including the grid lines of paper strips from the actual ECG signal. The rest dotted portion of the background noise are removed by component labeling [22]. Then a thinning algorithm [22] is applied on the two tone image to avoid repetition of co-ordinate information in the dataset (fig. 6). The pixel to pixel co-ordinate information is extracted and calibrated according to the electrocardiographic paper to generate an ASCII datafile. A time (in sec.) Vs. millivolt data-file is obtained for each of 12 lead ECG signal after processing as above [43]. The present database contains ECG from 85 normal and 85 diseased subjects, out of which 50 patients had acute myocardial infarction (MI) and rest 35 patients had Myocardial Ischemia.

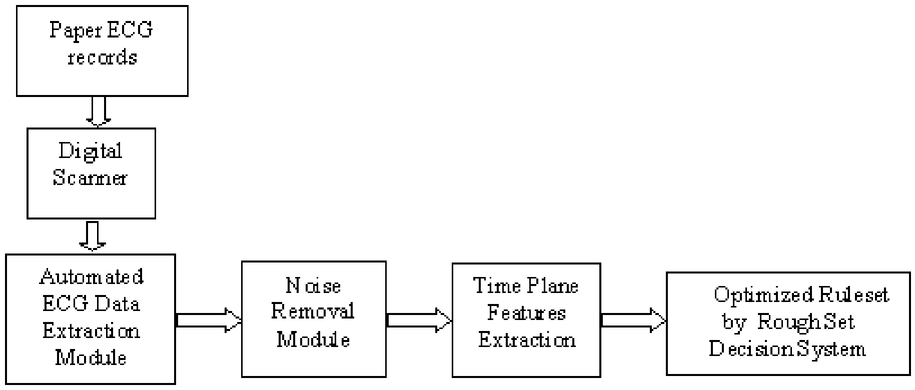

Paper ECG records
Digital Scanner
Automated ECG Data Extraction Module
Noise Removal Module
Time Plane Features Extraction
Optimized Ruleset by Rough Set Decision System

Fig. 5. Block Diagram of the Proposed System

Table 1. Extracted database of an image

Paper speed = 25 mm/s, Calibration factor = 10 mv/mm,
Total no. of points = 615, Heart rate = 85 beats/min

X (sec.)	Y (mv)	X (sec.)	Y (mv)	X (sec.)	Y (mv)	X (sec.)	Y (mv)	X (sec.)	Y (mv)
0.0032	0.36	0.3936	0.456	0.784	0.272	1.1744	0.32	1.5648	0.136
0.0064	0.36	0.3968	0.456	0.7872	0.272	1.1776	0.312	1.568	0.136
0.0096	0.36	0.4	0.456	0.7904	0.272	1.1808	0.312	1.5712	0.144
0.0128	0.36	0.4032	0.448	0.7936	0.272	1.184	0.312	1.5744	0.144
0.016	0.36	0.4064	0.448	0.7968	0.28	1.1872	0.304	1.5776	0.144
0.0192	0.352	0.4096	0.448	0.8	0.304	1.1904	0.304	1.5808	0.152
0.0224	0.352	0.4128	0.44	0.8032	0.368	1.1936	0.304	1.584	0.152
0.0256	0.352	0.416	0.432	0.8064	0.584	1.1968	0.304	1.5872	0.16
0.0288	0.352	0.4192	0.424	0.8096	0.624	1.2	0.304	1.5904	0.16
0.032	0.352	0.4224	0.416	0.8128	0.792	1.2032	0.296	1.5936	0.168
0.0352	0.344	0.4256	0.416	0.816	0.84	1.2064	0.296	1.5968	0.168
0.0384	0.344	0.4288	0.416	0.8192	0.928	1.2096	0.296	1.6	0.168
0.0416	0.344	0.432	0.408	0.8224	0.88	1.2128	0.296	1.6032	0.168
0.0448	0.344	0.4352	0.408	0.8256	0.848	1.216	0.296	1.6064	0.176
0.048	0.344	0.4384	0.4	0.8288	0.8	1.2192	0.288	1.6096	0.176
0.0512	0.336	0.4416	0.392	0.832	0.592	1.2224	0.288	1.6128	0.176
0.0544	0.336	0.4448	0.384	0.8352	0.456	1.2256	0.288	1.616	0.184
0.0576	0.328	0.448	0.384	0.8384	0.296	1.2288	0.288	1.6192	0.184
0.0608	0.328	0.4512	0.376	0.8416	0.176	1.232	0.28	1.6224	0.184
0.064	0.328	0.4544	0.376	0.8448	0.144	1.2352	0.28	1.6256	0.192
0.0672	0.32	0.4576	0.368	0.848	0.144	1.2384	0.28	1.6288	0.192
0.0704	0.32	0.4608	0.368	0.8512	0.144	1.2416	0.28	1.632	0.192
0.0736	0.32	0.464	0.36	0.8544	0.152	1.2448	0.28	1.6352	0.192
0.0768	0.32	0.4672	0.36	0.8576	0.152	1.248	0.28	1.6384	0.192
0.08	0.312	0.4704	0.36	0.8608	0.152	1.2512	0.28	1.6416	0.2
0.0832	0.312	0.4736	0.36	0.864	0.16	1.2544	0.28	1.6448	0.2
0.0864	0.304	0.4768	0.352	0.8672	0.168	1.2576	0.28	1.648	0.208
0.0896	0.304	0.48	0.352	0.8704	0.168	1.2608	0.28	1.6512	0.208
0.0928	0.304	0.4832	0.352	0.8736	0.168	1.264	0.28	1.6544	0.216
0.096	0.304	0.4864	0.344	0.8768	0.168	1.2672	0.28	1.6576	0.216
0.0992	0.296	0.4896	0.344	0.88	0.176	1.2704	0.28	1.6608	0.216
0.1024	0.296	0.4928	0.344	0.8832	0.184	1.2736	0.28	1.664	0.224
0.1056	0.296	0.496	0.336	0.8864	0.184	1.2768	0.28	1.6672	0.224
0.1088	0.288	0.4992	0.336	0.8896	0.184	1.28	0.28	1.6704	0.232
0.112	0.288	0.5024	0.328	0.8928	0.184	1.2832	0.28	1.6736	0.24
0.1152	0.288	0.5056	0.32	0.896	0.192	1.2864	0.28	1.6768	0.24
0.1184	0.28	0.5088	0.32	0.8992	0.192	1.2896	0.28	1.68	0.248
0.1216	0.296	0.512	0.32	0.9024	0.192	1.2928	0.272	1.6832	0.256
0.1248	0.344	0.5152	0.32	0.9056	0.2	1.296	0.272	1.6864	0.272

Table 1. (*continued*)

X (sec.)	Y (mv)	X (sec.)	Y (mv)	X (sec.)	Y (mv)	X (sec.)	Y (mv)	X (sec.)	Y (mv)
0.128	0.456	0.5184	0.32	0.9088	0.2	1.2992	0.272	1.6896	0.28
0.1312	0.616	0.5216	0.32	0.912	0.2	1.3024	0.272	1.6928	0.288
0.1344	0.68	0.5248	0.32	0.9152	0.208	1.3056	0.272	1.696	0.296
0.1376	0.784	0.528	0.32	0.9184	0.208	1.3088	0.272	1.6992	0.304
0.1408	0.824	0.5312	0.32	0.9216	0.208	1.312	0.272	1.7024	0.312
0.144	0.88	0.5344	0.32	0.9248	0.216	1.3152	0.264	1.7056	0.328
0.1472	0.944	0.5376	0.32	0.928	0.216	1.3184	0.264	1.7088	0.336
0.1504	0.776	0.5408	0.32	0.9312	0.216	1.3216	0.264	1.712	0.336
0.1536	0.68	0.544	0.32	0.9344	0.224	1.3248	0.264	1.7152	0.344
0.1568	0.568	0.5472	0.312	0.9376	0.224	1.328	0.264	1.7184	0.344
0.16	0.36	0.5504	0.312	0.9408	0.224	1.3312	0.264	1.7216	0.352
0.1632	0.216	0.5536	0.312	0.944	0.224	1.3344	0.264	1.7248	0.352
0.1664	0.168	0.5568	0.304	0.9472	0.232	1.3376	0.264	1.728	0.352
0.1696	0.168	0.56	0.304	0.9504	0.232	1.3408	0.264	1.7312	0.36
0.1728	0.168	0.5632	0.304	0.9536	0.232	1.344	0.264	1.7344	0.36
0.176	0.168	0.5664	0.304	0.9568	0.232	1.3472	0.264	1.7376	0.368
0.1792	0.168	0.5696	0.304	0.96	0.24	1.3504	0.264	1.7408	0.368
0.1824	0.176	0.5728	0.304	0.9632	0.248	1.3536	0.264	1.744	0.368
0.1856	0.176	0.576	0.304	0.9664	0.248	1.3568	0.264	1.7472	0.368
0.1888	0.176	0.5792	0.304	0.9696	0.256	1.36	0.264	1.7504	0.368
0.192	0.176	0.5824	0.304	0.9728	0.256	1.3632	0.264	1.7536	0.36
0.1952	0.176	0.5856	0.296	0.976	0.256	1.3664	0.264	1.7568	0.36
0.1984	0.184	0.5888	0.296	0.9792	0.256	1.3696	0.264	1.76	0.36
0.2016	0.184	0.592	0.296	0.9824	0.264	1.3728	0.272	1.7632	0.352
0.2048	0.192	0.5952	0.296	0.9856	0.264	1.376	0.272	1.7664	0.352
0.208	0.192	0.5984	0.296	0.9888	0.272	1.3792	0.28	1.7696	0.352
0.2112	0.2	0.6016	0.296	0.992	0.28	1.3824	0.28	1.7728	0.344
0.2144	0.208	0.6048	0.296	0.9952	0.288	1.3856	0.28	1.776	0.336
0.2176	0.208	0.608	0.296	0.9984	0.288	1.3888	0.28	1.7792	0.328
0.2208	0.208	0.6112	0.304	1.0016	0.296	1.392	0.28	1.7824	0.328
0.224	0.208	0.6144	0.304	1.0048	0.304	1.3952	0.28	1.7856	0.32
0.2272	0.216	0.6176	0.296	1.008	0.312	1.3984	0.28	1.7888	0.312
0.2304	0.216	0.6208	0.296	1.0112	0.312	1.4016	0.28	1.792	0.312
0.2336	0.224	0.624	0.296	1.0144	0.32	1.4048	0.28	1.7952	0.312
0.2368	0.224	0.6272	0.296	1.0176	0.328	1.408	0.272	1.7984	0.304
0.24	0.224	0.6304	0.296	1.0208	0.344	1.4112	0.272	1.8016	0.304
0.2432	0.224	0.6336	0.296	1.024	0.352	1.4144	0.272	1.8048	0.296
0.2464	0.232	0.6368	0.296	1.0272	0.36	1.4176	0.264	1.808	0.288
0.2496	0.232	0.64	0.296	1.0304	0.376	1.4208	0.264	1.8112	0.288
0.2528	0.232	0.6432	0.296	1.0336	0.384	1.424	0.264	1.8144	0.28
0.256	0.232	0.6464	0.296	1.0368	0.392	1.4272	0.264	1.8176	0.28

Table 1. (*continued*)

X (sec.)	Y (mv)	X (sec.)	Y (mv)	X (sec.)	Y (mv)	X (sec.)	Y (mv)	X (sec.)	Y (mv)
0.2592	0.24	0.6496	0.296	1.04	0.4	1.4304	0.264	1.8208	0.272
0.2624	0.24	0.6528	0.296	1.0432	0.408	1.4336	0.256	1.824	0.272
0.2656	0.24	0.656	0.296	1.0464	0.416	1.4368	0.256	1.8272	0.264
0.2688	0.248	0.6592	0.296	1.0496	0.416	1.44	0.256	1.8304	0.264
0.272	0.248	0.6624	0.296	1.0528	0.424	1.4432	0.256	1.8336	0.264
0.2752	0.248	0.6656	0.296	1.056	0.432	1.4464	0.248	1.8368	0.264
0.2784	0.248	0.6688	0.296	1.0592	0.432	1.4496	0.248	1.84	0.264
0.2816	0.248	0.672	0.296	1.0624	0.44	1.4528	0.24	1.8432	0.256
0.2848	0.248	0.6752	0.296	1.0656	0.44	1.456	0.24	1.8464	0.256
0.288	0.256	0.6784	0.296	1.0688	0.448	1.4592	0.24	1.8496	0.248
0.2912	0.256	0.6816	0.296	1.072	0.448	1.4624	0.232	1.8528	0.248
0.2944	0.256	0.6848	0.296	1.0752	0.448	1.4656	0.232	1.856	0.248
0.2976	0.264	0.688	0.296	1.0784	0.448	1.4688	0.232	1.8592	0.248
0.3008	0.264	0.6912	0.304	1.0816	0.44	1.472	0.232	1.8624	0.248
0.304	0.272	0.6944	0.304	1.0848	0.44	1.4752	0.24	1.8656	0.24
0.3072	0.28	0.6976	0.304	1.088	0.44	1.4784	0.288	1.8688	0.24
0.3104	0.28	0.7008	0.312	1.0912	0.432	1.4816	0.408	1.872	0.24
0.3136	0.296	0.704	0.32	1.0944	0.424	1.4848	0.488	1.8752	0.24
0.3168	0.304	0.7072	0.32	1.0976	0.424	1.488	0.64	1.8784	0.24
0.32	0.312	0.7104	0.312	1.1008	0.416	1.4912	0.688	1.8816	0.24
0.3232	0.32	0.7136	0.312	1.104	0.416	1.4944	0.816	1.8848	0.24
0.3264	0.32	0.7168	0.312	1.1072	0.408	1.4976	0.728	1.888	0.24
0.3296	0.328	0.72	0.312	1.1104	0.408	1.5008	0.688	1.8912	0.232
0.3328	0.336	0.7232	0.312	1.1136	0.4	1.504	0.616	1.8944	0.232
0.336	0.344	0.7264	0.304	1.1168	0.392	1.5072	0.464	1.8976	0.232
0.3392	0.36	0.7296	0.304	1.12	0.392	1.5104	0.248	1.9008	0.232
0.3424	0.368	0.7328	0.304	1.1232	0.384	1.5136	0.136	1.904	0.232
0.3456	0.376	0.736	0.304	1.1264	0.376	1.5168	0.104	1.9072	0.232
0.3488	0.376	0.7392	0.304	1.1296	0.368	1.52	0.088	1.9104	0.224
0.352	0.384	0.7424	0.296	1.1328	0.368	1.5232	0.08	1.9136	0.224
0.3552	0.4	0.7456	0.296	1.136	0.368	1.5264	0.088	1.9168	0.224
0.3584	0.408	0.7488	0.296	1.1392	0.36	1.5296	0.088	1.92	0.224
0.3616	0.416	0.752	0.296	1.1424	0.36	1.5328	0.096	1.9232	0.224
0.3648	0.416	0.7552	0.296	1.1456	0.352	1.536	0.096	1.9264	0.224
0.368	0.424	0.7584	0.296	1.1488	0.344	1.5392	0.104	1.9296	0.224
0.3712	0.432	0.7616	0.296	1.152	0.336	1.5424	0.112	1.9328	0.224
0.3744	0.44	0.7648	0.296	1.1552	0.336	1.5456	0.112	1.936	0.224
0.3776	0.44	0.768	0.288	1.1584	0.328	1.5488	0.12	1.9392	0.224
0.3808	0.448	0.7712	0.288	1.1616	0.328	1.552	0.12	1.9424	0.216
0.384	0.448	0.7744	0.28	1.1648	0.328	1.5552	0.128	1.9456	0.216
0.3872	0.456	0.7776	0.28	1.168	0.32	1.5584	0.128	1.9488	0.216

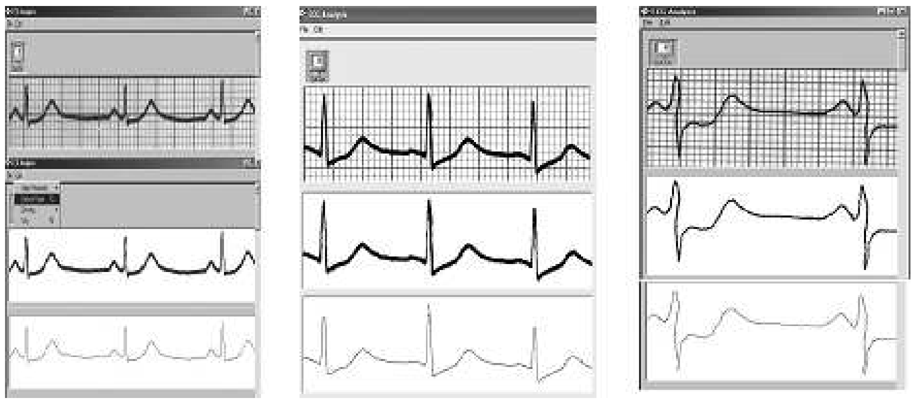


Fig. 6. Original ECG image[Upper], ECG signal after removal of noise[Middle], ECG signal after thinning [Lower]

The ASCII datafile generated for the image of ischemic sample shown in fig. 6, is given in table 1.

5.2 Removal of Noises from ECG Signals

Electrocardiographic signals may be corrupted by different types of noises [18]. Typical examples are: 1.power line Interference, 2.electrode contact noise, 3.motion artifacts, 4.muscle contraction(electrmyographic,EMG), 5.baseline drift and ECG amplitude modulation with respiration, and 6.electrosurgical noise. All the noises are simulated by a software package Cool Edit Pro offered by Syntrillium Software Corporation. This is done to get a realistic situation for the algorithm. The EMG is simulated by adding random noise(white noise) to the ECG. An FIR filter depending upon Savitzky-Golay algorithm is developed to remove EMG like white noises from the ECG signals. 50Hz sinusoid is modeled as power line interference and added with ECG. The base line drift due to respiration was modeled as a sinusoid of frequency 0.15 to 0.4 Hz. A 50 Hz Notch filter is designed for rejection of frequency band due to power line oscillation. The selection of notch width is very important. It should not affect the notch depth and hence, would not be too narrow as well as too wide. In our experiment, the notch width is fixed around 4 to 6 Hz keeping 50 Hz at the center of the notch. So the signal should not be distorted so much after using notch filter, especially,the low frequency band should remain less affected because this band carries more useful information.

Then a high pass filter of critical frequency 0.6 Hz is developed to block the low frequency noise signal that causes the base line shift. The fundamental frequency of ECG signals generally varies from 0.8 Hz to 1.8 Hz and the ECG bandwidth is 0.8Hz to 500Hz. But in conventional ECG machines this bandwidth is reduced to 0.8Hz to 80Hz since the mechanical stylus of ECG machines cannot move faster. Hence, 0.6Hz high pass filter blocks only the noise and pass the original signal.

Both these FIR filters are also designed by the Cool Edit Pro software. The abrupt base line shift is simulated by adding a dc bias for a given segment of the ECG. This noise can be blocked with the help of the high pass filter described above.

Since motion artifact is similar to baseline drift in respiration, it was not specifically modeled. All of these noises are added to the ECG signal to simulate the composite noise. This corrupted ECG signal is passed through all the filters described above to get almost noise free ECG signal. All types of noise levels are varied from 10% to 30% and the generated filters have produced good response in all the cases.

5.3 Time-Plane Features Extraction

Accurate detection of the R-R interval between two consecutive ECG waves is very important to extract the time based features from ECG signals. For this purpose, the 2^{nd} order derivative of the captured signal is computed by 5-point Lagrangian interpolation formula for differentiation [27] given below :

$$f_0' \quad = \quad \frac{1}{12h}(f_{-2} - 8f_{-1} + 8f_1 - f_2) + \frac{h^4}{30}f^v(\xi). \tag{6}$$

ξ lies between the extreme values of the abscissas involved in the formula. After squaring the values of 2^{nd} order derivative, a square-derivative curve having only high positive peaks of small width at the QRS complex region can be obtained (fig. 7). A small window of length (say W) was taken to detect the area of this curve and we obtained maximum area at those peak regions.

The local maxima of these peak regions are considered as R-peak. For this experiment the value of W is set as ~0.07 sec. The system is tested for both noise free and noisy signals. The levels of all type of noises are increased from 0% to 30% and still we achieved 99.5% accuracy in detection of QRS complexes.

In order to accurately detect P wave and ST segments, the isoelectric line must be correctly identified. Most methods employed for this purpose are based on the assumption that the isoelectric level of the signal lies on the area ~80 ms left of the R-peak, where the first derivative becomes equal.

In particular, let y_1,y_2, ..., y_n be the samples of a beat [R-R interval], $y_1\prime$, $y_2\prime$, ..., $y_n\prime_{-1}$ be their first differences and y_r the sample where the R-peak occurs. The isoelectric level samples y_b are then defined if either of the two following criteria is satisfied:

$$\left| y\prime_{r-j-int(0.08f)} \right| = 0, \quad j = 1, 2, \ldots, 0.01f \quad or \tag{7}$$

$$| \ y\prime_{r-j-int(0.08f)} | \quad \leq \quad | \ y\prime_{r-i-int(0.08f)} \ | \ , \ i, \ j \ = \ 1,2,\ldots,0.02f$$

where $f$is the sampling frequency. After detection of baseline, the location of P wave is determined from the first derivative of the samples.

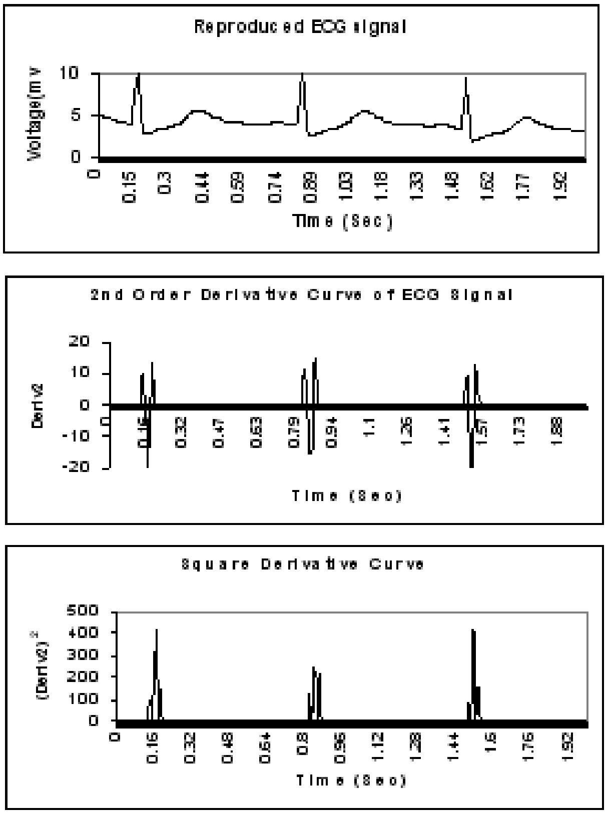


Fig. 7. QRS complex or R-R interval Detection

The R wave can be detected very reliably and for this reason, it is used as the starting point for ST segment [fig 8] processing, and for T wave detection. In most algorithms dealing with ST segment processing it is assumed that the ST segment begins at 60 ms after the R-peak in normal sinus rhythm. In the case of tachycardia (RR-interval <600 ms), the beginning of the ST segment is marked at 40 ms after the R peak. The ST-segment duration has beat-to-beat variability, but since this is not easily determined, many algorithms assume that ST has a predefined length of 160 ms (this means that the end point is 220 ms after R-peak in the normal case and 200 ms otherwise).

Other algorithms follow the Bazzet formula [39], that links the ST segment duration with the RR interval duration. The above mentioned ST segment limits are in general agreement with the recommendation the European ST-T database and with the observations in [28, 40, 64].

Our algorithm adopted the first assumption and once the beginning of ST segment is detected, it computes the slope and also detects the zero crossings (basically isoelectric level crossing). Depending on the zero crossings and shape of each wave, a syntactic approach [47] is developed for the detection of P, Q,

Table 2. A segment of extracted ECG features

Heart Rate bpm	PR Interval Sec.	P Wave Height mm	P wave Width Sec.	QRS Width Sec.	QT Interval Sec.	RR Interval Sec.	QTc	QRS Voltage mm
79	0.16	0.2	0.08	0.08	0.4	0.76	0.09	3
52	0.14	1.07	0.08	0.12	0.48	1.12	0.09	15
98	0.24	1.43	0.08	0.12	0.32	0.6	0.08	8
60	0.1	0.8	0.04	0.08	0.48	1	0.09	4
93	0.16	1	0.04	0.08	0.4	0.6	0.10	5
68	0.2	2.34	0.08	0.04	0.36	0.88	0.07	13
94	0.2	1.12	0.08	0.06	0.32	0.56	0.08	8
114	0.12	2.15	0.04	0.04	0.32	0.52	0.08	14
122	0.16	2.12	0.08	0.04	0.32	0.48	0.09	14
72	0.16	2.36	0.08	0.04	0.32	0.84	0.07	21
100	0.24	1.5	0.04	0.04	0.36	0.6	0.09	10
83	0.12	1.05	0.08	0.04	0.4	0.72	0.09	13
88	0.16	1.62	0.08	0.04	0.4	0.68	0.09	11
100	0.16	-0.2	0.04	0.08	0.36	0.64	0.09	3
79	0.17	1.2	0.08	0.04	0.36	0.76	0.08	7
56	0.24	1	0.08	0.04	0.36	1.08	0.07	3
79	0.16	0.2	0.04	0.04	0.36	0.76	0.08	8
83	0.24	-0.1	0.02	0.08	0.4	0.72	0.09	7
57	0.24	1	0.08	0.04	0.36	1.04	0.07	7
94	0.2	1	0.04	0.08	0.32	0.64	0.08	5
58	0.16	1	0.08	0.04	0.44	1.04	0.08	4
68	0.16	1	0.08	0.04	0.32	0.92	0.06	26
77	0.16	1	0.08	0.04	0.36	0.76	0.08	4
86	0.16	1	0.08	0.04	0.36	0.68	0.08	11
112	0.16	2	0.08	0.04	0.32	0.52	0.09	11
104	0.16	1	0.08	0.04	0.32	0.6	0.08	12
102	0.16	1	0.04	0.04	0.32	0.6	0.08	19
134	0.2	2	0.08	0.04	0.28	0.6	0.07	7
90	0.16	2	0.08	0.04	0.36	0.72	0.08	15
79	0.16	1	0.08	0.04	0.36	0.72	0.08	8
90	0.16	2	0.08	0.04	0.32	0.72	0.07	23

R, S and T waves and for the computation of different attributes of those waves (fig. 8). For getting QRS complex we achieved 99.5% accuracy, while for T waves the accuracy was 96.7% and for P waves the accuracy obtained was 92.2%. A part of the extracted features are given in table 2.

5.4 Development of Knowledge Base

A knowledge base regarding ECG interpretation is also developed using the opinion of the reputed cardiologists of hospitals and clinical centers. For this purpose we selected 20 doctors and gave them different sample questions about

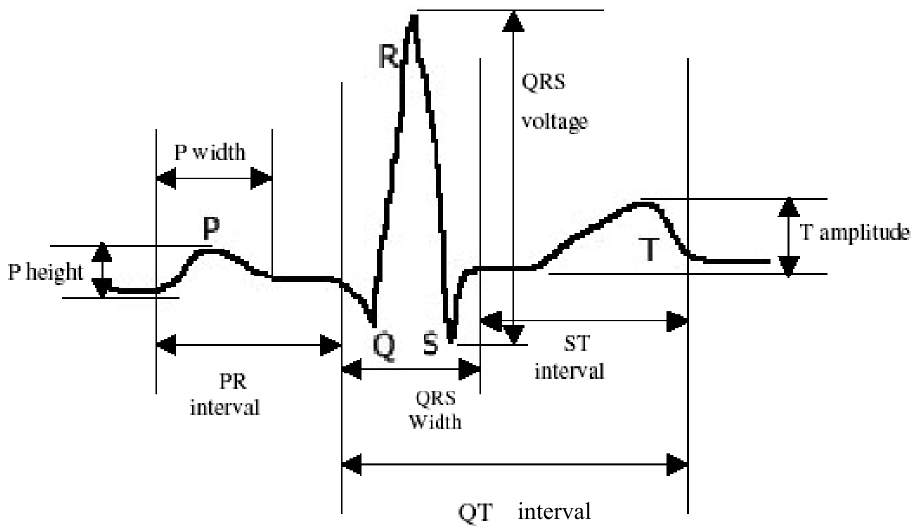


Fig. 8. Different time plane features of one cycle of ECG

ECG interpretation. From their feedback and after consultation of some medical books [4, 24, 26] we have selected 12 time plane features for disease identification. They are listed below:

1.Heart Rate (60-100 beats/min), 2. PR interval (0.12-0.2s), 3.P wave height (2.5mm), 4. P wave width (<0.12s), 5. QRS width (0.1or < 0.1 s), 6. QRS voltage (35mm), 7.QTc= (QT interval/ Sqrt RR interval) (<=0.44s), 8. Abnormal Q wave, 9. R wave Progression, 10. ST segment, 11. Reciprocity in T wave, and 12. T wave.

Among these 12 features, the first 9 are numerical attributes whereas the last three are categorical attributes. The normal ranges of all the numerical attributes are given in brackets except for number 8 and 9. The Q wave is generally abnormal if its duration is 0.04 second or more in lead I, all three inferior leads (II, III, AV_F), or the leads V_3 to V_6. On the other hand, a normal R wave progression can be achieved by observing a gradual increase of R/S ratio in chest leads. So, small R or absence of it in chest leads cause an abnormal or poor R wave progression.

Among the categorical attributes, the ST segment has three categories (i) Isoelectric, (ii) Elevated and (iii) Depressed. The T wave has two categories (i) Positive and (ii) Negative/ Inverted. The reciprocity in T wave can be examined by checking the category of ST segment in specific leads. The anterior and inferior leads tend to show inverse patterns. Thus, inferior leads show ST segment elevation, with reciprocal ST depression, often seen in anterior leads.

These 12 features are adequate for separation of normal and abnormal ECG patterns. We have computed one of the most important rough set quantitative measure called degree of dependency (k) to find the most significant conditional attributes with respect to the decision attributes. These attributes are arranged according to their significance. The detailed calculations and inferences are given in result section.

For classification of different diseases, the findings of the signal abnormalities at different lead positions take an important role.

5.5 Development of Inference Engine

A rule-based rough-set decision system is generated for the development of an inference engine for disease identification from the time-plane feature analysis of ECG signals. One of popular and widely used, although slightly outdated, rough-set software tool box is ROSETTA [45, 57][1]. This software supports different options of generating decision tables, reducts, discretization techniques, decision algorithms and classifications. We have used this software for our experiment. Learning samples are processed in the following way. First, a knowledge base is acquired for the data set i.e, for normal and abnormal data regarding both quantitative and qualitative natures of those extracted time-plane features. Knowledge base consists of objects, which are represented using conditional attributes and decision parameters. All the time plane features described above get their specific attributes according to knowledgebase and used as the input parameters of the decision tables, a portion of which is given in table 3 and 4. Here, two consecutive decision tables are framed for the generation of two sets of rules. The first set is for separation of normal and abnormal data and the second one is for the classification of diseased data set (specially MI).

Consequently, the acquired data are quantized to convert real attribute values into discretized form, allowing further rule processing. Based on the discrete values, attributes are analyzed in terms of discernibility investigation. Sets of attributes allowing partition of object classes are then revealed. These sets are called reducts.

The ROSETTA system supports a variety of quantization as well as reduct and rule generation procedures. However, the details of these lie beyond the scope of this report. For our experiments the following processing parameters were employed:

- Equal frequency binning using 3 intervals is used for discretization.
- Object related genetic algorithm producing a set of rules via minimal attribute subsets that discern object classes; reducts and rules are generated upon analysis of all learning patterns.

These processing parameters were chosen during a preliminary research aimed at optimizing the system efficiency and generation ability.

6 Experimental Results

6.1 Rule Generation

In this experiment two consecutive rule sets are generated from two separate decision tables, portions of which are given in table 3 and 4, respectively.

[1] Downloadable at `http://rosetta.lcb.uu.se/`

Table 3. A portion of decision table 1

Heart Rate	PR Interval	P Wave height	P wave width	QRS width	QTc	QRS voltage	R Wave Prog	Abn. Q waves	ST Segment	Reciprocity	T waves	Disease
Conditional attributes												
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	Decision class
String	String	String	String	String	String	String	String	String	String	String	String	String
N	N	N	N	N	N	N	A	P	E	P	+	MI
B	N	N	N	A	N	N	A	P	E	P	+	MI
N	A	N	N	A	N	N	A	P	E	Ab	-	MI
N	N	N	N	N	N	N	A	P	D	Ab	-	MI
B	A	N	N	N	N	N	A	P	E	Ab	+	MI
N	N	N	N	N	N	N	N	Ab	I	Ab	+	N
N	N	N	N	N	N	N	N	Ab	I	Ab	+	N
T	N	N	N	N	N	N	N	Ab	I	Ab	+	N
T	N	N	N	N	N	N	N	Ab	I	Ab	+	N
N	A	N	N	N	N	N	N	Ab	E	Ab	+	ISC
N	N	N	N	N	N	N	N	Ab	E	Ab	+	ISC
N	N	N	N	N	N	N	A	Ab	E	Ab	+	ISC
N	N	N	N	N	N	N	A	Ab	E	Ab	+	ISC
N	N	N	N	N	N	N	A	Ab	E	Ab	+	ISC
B	N	N	N	N	N	N	N	Ab	E	Ab	+	N

Table 3 is generated by different time plane features after acquiring specific string attribute, depending on the quantitative measures obtained from the generated knowledgebase. For example, if the heart rate of a patient is within normal range then it will get a string attribute 'N'. If it falls below the lower limit of the normal range then it will obtain another attribute 'B' for Bradycardia, whereas if it is higher than the upper limit of the normal range then the attribute will be 'T' for Tachycardia. The meaning of the other attributes are given at lower part of the table 3.

According to the opinion of cardiologists and different medical books, all the leads cannot show abnormal patterns for a specific disease. Only a specific group of leads may show a particular abnormality for a particular disease. Hence, we have to find that exact lead combination for that meticulous disease. Table 4 is generated mainly according to the lead positions where the specific abnormality is present. The string attributes in this table 4 are basically the lead combination where the specific abnormal feature (conditional attribute) has been found. For example, for the first case of table 4, the disease is Anterior type MI and the first column of that row represents the lead combination L3, V1, V2, V3 where the elevated ST segment has been found and so on. The bipolar limb leads are represented as L1, L2, L3, the monopolar limb leads (augmented) are represented as VR, VL, VF and the chest leads are represented as V1 – V6. Intuitively, a "strong" rule is both accurate and has a high coverage. The accuracy of a rule reflects how trustworthy its consequent is. Snapshots of the generated rule set from table 3 and the corresponding confusion matrix, generated using standard voting classifier are given in figures 9 and 10. We consider both LHS and RHS coverage factor for the selection of the optimum rule set. For example, rule 1 of fig. 9 gives the decision according to LHS coverage factor that only 31.4% patients having ECG where Abnormal Q wave is present(P) do

Table 4. A portion of decision table 2

ST Segment Elevated	ST Segment Depressed	T wave inverted	T wave Asymmetric	T wave Symmetric	Hyper-acute T Present	Pathologic Q Present	Disease
String	String	String	String	String	String	String	String
L3,V1,V2,V3	L1,L2,VR,VL, VF,V4	L1,L2,VR,VL, VF,V4,	V2,V3	L1,L2,L3,VR, VL,VF	V1,V2,V3,V4	VR,VL,V5	AN
L1,L2,VL,V2, V3,V4	L3,VR,V1	L3,VR,V1,V4, V5,V6	L2,VF,V1,V2, V3,V4	L1,L3,VR,VL, V5,V6	VL,VF,V1	L2,VR,VF	AN
L1,L2,VL,V1, V2,V3	L3,VR,VL,VF	L3,VR,VL,VF	L1,L2,VR,VL, VF,V1	L1,L3,V4	V3	L2,V2,V3	IN
L1,VL,V1,V2, V3,V4	L2,L3,VR,VF	L2,L3,VR,VF	VR,V6	L1,L2,L3VL, VF,V1	L1,L3,VL,V2, V3,V4	L2,L3,VF,V5	IN
L1,VL,V2,V2, V3,V4	L2,L3,VR,VF	L2,L3,VR,VF	L1,L3,VR,VF	L2,VL,V1,V2, V3,V4	N	L1,L2,L3,VR, VF,V1,V2,V3	IL
L1,L2,VL,V1, V2,V3	L3,VR,VF	L3,VR,VF,V4	L2,VR,V3,V4, V5,V6	L1,L3,VL,VF, V1,V2	V1,V2	L1,L3,VR,VL, V4,V5,V6	IL
L1,L2,VL,V2, V3,V4	L3,VR,VF,V5, V6	L3,VR,VF,V5, V6	L1,L2,VR	L3,VL,VF,V1, V2,V3	V2,V3	L1,L2,L3,VL, VF,V1,V2,V3	AS
L1,L2,VL,V1, V2,V3	L3,VR	L3,VR,VF	L3,VR,VL, VF,V3,V5	L1,L2,V1, V2,V5,V6	L3,VR,V3, V4,V5	L1,L2,L3,VF, V4,V5,V6	AS

Table 5. Result obtained from rule based rough set decision system

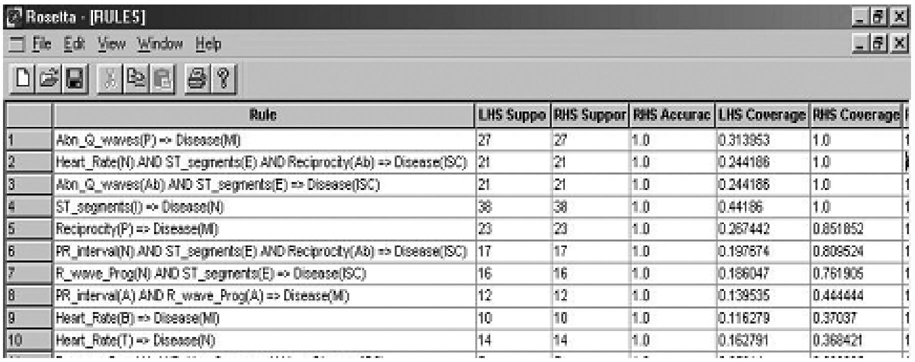
Type of Samples	No. of Trained Samples	No. of Untrained Samples	Accuracy for Trained Samples	Accuracy for Untrained Samples
Normal	38	47	100% (38/38)	100% (47/47)
Ischemia	21	14	100% (21/21)	93% (13/14)
MI	27	23	100% (27/27)	100% (23/23)

suffer from Myocardial Infarction(MI). Whereas from the inverse decision rule, considering RHS coverage factor, it can be concluded that 100% patients suffering from MI have ECG where abnormal Q wave is present. So, inverse decision rule provide more strong explanation of the generated decision. Obviously, rule 4 having highest LHS and RHS coverage factor will be the strongest. The first 7 rule sets with high accuracy and coverage factor (both LHS and RHS) are taken for the separation of normal and diseasesd data set. Both trained and untrained samples for all the three sets of dataset (e.g. Normal, Ischemia and Myocardial Infarction) are fed to the Inference system and the result obtained is given in table 5. The numbers given in brackets in table 5 represent the number of properly classified samples versus all tested samples. The confusion matrix (fig. 10) shows cent percent accuracy for all the three set of trained data. Table 5 supports this prediction. Still, the present system is tested by three types of ECG data samples and encouraging result is obtained.

The rule set from table 4 are generated in the same manner as described above and is shown in figure 11. The optimum rule set is chosen analogously by considering the LHS and RHS coverage and stabilty factors for classification of MI according to the location of the left ventricular cone where infarction actually occurs. MI can be classified as Anterior(AN), Inferior(IN), Antero-lateral(AL), Infero-lateral(IL), Antero-septal(AS), Infero-septal(IS) etc. The number of data for training and testing for all classes of MI is not adequate at present. Since,

Table 6. Result obtained by comparison with 10 fold cross validation study

Number of Folds	For **see 5** algorithm		For rough set	
	Number of rules	Rate of accuracy (%)	Number of rules	Rate of accuracy (%)
0	3	70	7(25)	100
1	5	90	8(25)	100
2	3	80	8(25)	100
3	4	90	8(25)	100
4	5	100	10(38)	93
5	5	90	7(25)	100
6	4	70	8(25)	100
7	5	100	7(24)	100
8	4	90	8(25)	100
9	5	80	7(30)	98



	Rule	LHS Suppo	RHS Suppor	RHS Accurac	LHS Coverage	RHS Coverage
1	Abn_Q_waves(P) => Disease(MI)	27	27	1.0	0.313953	1.0
2	Heart_Rate(N) AND ST_segments(E) AND Reciprocity(Ab) => Disease(ISC)	21	21	1.0	0.244186	1.0
3	Abn_Q_waves(Ab) AND ST_segments(E) => Disease(ISC)	21	21	1.0	0.244186	1.0
4	ST_segments(I) => Disease(N)	38	38	1.0	0.44186	1.0
5	Reciprocity(P) => Disease(MI)	23	23	1.0	0.267442	0.851852
6	PR_interval(N) AND ST_segments(E) AND Reciprocity(Ab) => Disease(ISC)	17	17	1.0	0.197674	0.809524
7	R_wave_Prog(N) AND ST_segments(E) => Disease(ISC)	16	16	1.0	0.186047	0.761905
8	PR_interval(A) AND R_wave_Prog(A) => Disease(MI)	12	12	1.0	0.139535	0.444444
9	Heart_Rate(B) => Disease(MI)	10	10	1.0	0.116279	0.37037
10	Heart_Rate(T) => Disease(N)	14	14	1.0	0.162791	0.368421

Fig. 9. A Portion of Generated Rule Set from table 1

the dataset contain good number of data of classes AN, IN, AS and IL our classification scheme on these four classes only. In future, the system will be tested by more number of samples and few other types of diseases.

The result is compared with a standard classification algorithm See5 (http://www.rulequest.com) which is based on the ID3 algorithm [59] and is the successor of the C4.5 program [60]. A 10 fold cross validation study for computational comparison, which include classification accuracy and number of rules, generated is given in table 6. The rough set tool box ROSETTA has generated 24 to 38 rules (shown in bracket in table 6) for each fold but we have taken the rules only having RHS coverage greater than 0.4. Actual classification was done by those reduced rule set of 6 to 10 elements. Hence, in rough set approach much better result can be obtained in cost of more number of rules.

6.2 Calculation of Degree of Dependency (k)

Let us consider the information system given in table 3 where U $=\{x_i$: i= 1,2,3,...,15$\}$ and C = $\{c_j$: j= 1,2,3,.....,11,12,13$\}$.

The following partitions can be obtained form the information system table 3:

$P_{c1}= \{\{x_2, x_5, x_{15}\}, \{x_8, x_9\}, \{x_1, x_3, x_4, x_6, x_7, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}\}\}$
$P_{c2}= \{\{x_3, x_5, x_{10}\}, \{x_1, x_2, x_4, x_6, x_7, x_8, x_9, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\}\}$
$P_{c5}= \{\{x_2,x_3\}, \{x_1,x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\}\}$
$P_{c8}= \{\{x_1, x_2, x_3, x_4, x_5, x_{12}, x_{13}, x_{14}\}, \{x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{15}\}\}$
$P_{c9}= \{\{x_1, x_2, x_3, x_4, x_5\}, \{x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\}\}$
$P_{c10}= \{\{x_4\}, \{x_1, x_2, x_3, x_5, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\}, \{x_6, x_7, x_8, x_9\}\}$
$P_{c11}= \{\{x_1, x_2\}, \{x_3,x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\}\}$
$P_{c12}= \{\{x_3,x_4\}, \{x_1, x_2, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\}\}$
$P_{c13}= \{\{x_1, x_2, x_3, x_4, x_5\}, \{x_6, x_7, x_8, x_9, x_{15}\}, \{x_{10}, x_{11}, x_{12}, x_{13}, x_{14}\}\}$

Here we excluded the conditional attributes C_3, C_4 and C_6, C_7, since all the samples are indiscernible with respect to these conditions. Hence, out of 12 attributes 8 attributes are now considered.

According to equation 5,

k = γ (C, D) = $\frac{|POS_C(D)|}{|U|}$, where, POS_C (D) = $\bigcup_{X\in U/D} C_*(X)$ is known as a positive region of the partition U/D with respect to C.

If we consider C=$\{C_{12}\}$ and D=$\{C_{13}\}$ then,

$$k = \frac{|\{x_3,x_4\}|}{|\{x_1,x_2,x_3,......,x_{15}\}|}.= 2/15 = 0.13$$

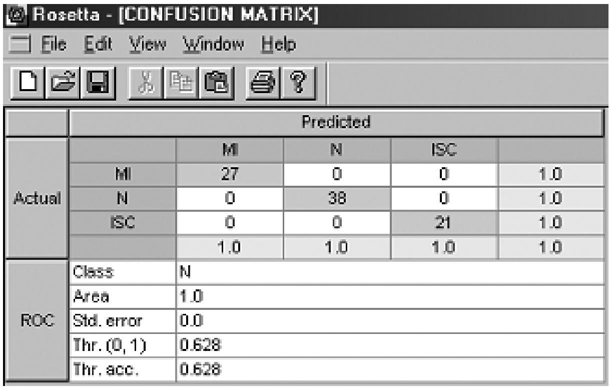
Now, if C=$\{C_{12}, C_{11}\}$ and D=$\{C_{13}\}$ then

U/I(C) = U/I(C_{12}, C_{11}) = $P_{c12\otimes}P_{c11}= \{\{x_1, x_2\}, \{x_3,x_4\}, \{x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\}\}$

$$k = \frac{|\{x_1,x_2,x_3,x_4\}|}{|\{x_1,x_2,x_3,......,x_{15}\}|}.= 4/15 = 0.26.$$

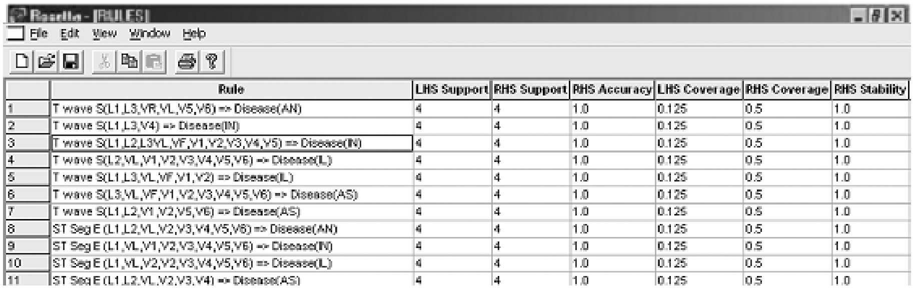
Hence, it is noted that the dependency of D on C has doubly incremented by considering both C_{12} and C_{11} instead of considering only C_{12}. In a similar manner, if the number of conditional attributes are gradually increased by the order given below, it can be noted that the value of k also increases and becomes 1 when all the 8 attributes are considered. Different values of k for different conditions are given below:

if C=$\{C_{12}, C_{11},C_9\}$,	k = 5/15 = 0.33
if C=$\{C_{12}, C_{11},C_9,C_8\}$,	k = 8/15 = 0.53
if C=$\{C_{12}, C_{11},C_9,C_8,C_5\}$,	k = 8/15 = 0.53
if C=$\{C_{12}, C_{11},C_9,C_8,C_5, C_{10}\}$,	k = 12/15 = 0.80
if C=$\{C_{12}, C_{11},C_9,C_8,C_5, C_{10}, C_2\}$,	k = 13/15 = 0.87
if C=$\{C_{12}, C_{11},C_9,C_8,C_5, C_{10}, C_2, C_1\}$,	k = 15/15 = 1.00



		Predicted			
		MI	N	ISC	
Actual	MI	27	0	0	1.0
	N	0	38	0	1.0
	ISC	0	0	21	1.0
		1.0	1.0	1.0	1.0
ROC	Class	N			
	Area	1.0			
	Std. error	0.0			
	Thr. (0, 1)	0.628			
	Thr. acc.	0.628			

Fig. 10. Confusion Matrix Output for Standard Voting



	Rule	LHS Support	RHS Support	RHS Accuracy	LHS Coverage	RHS Coverage	RHS Stability
1	T wave S(L1,L3,VR,VL,V5,V6) => Disease(AN)	4	4	1.0	0.125	0.5	1.0
2	T wave S(L1,L3,V4) => Disease(IN)	4	4	1.0	0.125	0.5	1.0
3	T wave S(L1,L2,L3VL,VF,V1,V2,V3,V4,V5) => Disease(IN)	4	4	1.0	0.125	0.5	1.0
4	T wave S(L2,VL,V1,V2,V3,V4,V5,V6) => Disease(IL)	4	4	1.0	0.125	0.5	1.0
5	T wave S(L1,L3,VL,VF,V1,V2) => Disease(IL)	4	4	1.0	0.125	0.5	1.0
6	T wave S(L3,VL,VF,V1,V2,V3,V4,V5,V6) => Disease(AS)	4	4	1.0	0.125	0.5	1.0
7	T wave S(L1,L2,V1,V2,V5,V6) => Disease(AS)	4	4	1.0	0.125	0.5	1.0
8	ST Seg E (L1,L2,VL,V2,V3,V4,V5,V6) => Disease(AN)	4	4	1.0	0.125	0.5	1.0
9	ST Seg E (L1,VL,V1,V2,V3,V4,V5,V6) => Disease(IN)	4	4	1.0	0.125	0.5	1.0
10	ST Seg E (L1,VL,V2,V2,V3,V4,V5,V6) => Disease(IL)	4	4	1.0	0.125	0.5	1.0
11	ST Seg E (L1,L2,VL,V2,V3,V4) => Disease(AS)	4	4	1.0	0.125	0.5	1.0

Fig. 11. A Portion of Generated rule set from table 2

Inclusion of C_{10} significantly changes the value of k. So, it should be the most significant condition. Where as effect of C_5 is null and so it is least significant. According to the significance these attributes can be arranged serially as $C_{10,}C_8$, C_{11}, C_{12}, C_1, C_9, C_2, C_5.

Where, N$\rightarrow$ Normal, A$\rightarrow$ Abnormal, P$\rightarrow$ Present, AB$\rightarrow$ Absent, B$\rightarrow$ Bradycardia, T$\rightarrow$ Tachycardia, E$\rightarrow$ Elevated, D$\rightarrow$ Depressed, I$\rightarrow$ Isoelectric, MI$\rightarrow$ Myocardial Infarction, ISC$\rightarrow$ Ischemia.

7 Conclusion

In this paper the rough set decision system is used for two phase classification of ECG signals. To do so, an automated off-line data acquisition package is developed to extract the ECG signals from paper records. Six different types of noises may corrupt those extracted ECG signals. So, different smoothing and filtering techniques are adopted for making those signals almost noise free. A knowledge base about the time plane features and ECG interpretation is developed from various medical books and from the feed back of different reputed cardiologists. The time-plane features of ECG signals are extracted from each of the 12 lead

ECG signals with the help of differentiation and syntactic approaches. A rule-based rough set decision system is developed from these time-plane features to make an inference engine for two phase ECG classification.

Computation of degree of dependency (k), a rough set quantitative measure, has also been done to arrange the set of attributes according to their significance in order to make the decision. Apart from that, a computational comparison has been done by using C5 algorithm which is developed on the basis of ID3 algorithm and is a higher version of C 4.5 algorithm. A 10 fold cross validation study has been prepared for this comparison. It has been noted that rough set approach is quite encouraging in this field of study.

At present, the system is tested with three types of ECG data- Normal, Myocardial Ischemia and Myocardial Infarction. In the first phase the ischemia and MI data are separated from normal data set and in the next phase it can classify the MI data according to the location of the left ventricular cone where infarction actually occurs. In future, the system will be tested with large number and different types of dataset.

Acknowledgements

The part of the work is funded by Department of Bio-technology (DBT) , Govt. of India.

References

1. Abreu-Lima, C., de Sa, J.P.: Automatic classifiers for the interpretation of electrocardiograms. Rev. Port. Cardiol. 17(5), 415–428 (1998)
2. al-Fahoum, A.S., Howitt, I.: Combined wavelet transformation and radial basis neural networks for classifying life-threatening cardiac arrhythmias. Med. Biol. Eng. Comput. 37(5), 566–573 (1999)
3. Al-Nashash: Cardiac Arrhythmia Classification using Neural Networks. Technology and Healthcare: Official Journal of the European Society for Engineering and Medicine 8(6), 363–372 (2000)
4. Goldberger, A.L.: Clinical Electrocardiography, A Simlified Approach, 6th edn., Harcourt India Pvt. Ltd.,
5. Bhullar, H.K., de Bono, D.P., Fothergill, J.C., Jones, N.B.: A computer based system for the study of QT intervals. In: Proceedings Computers in Cardiology, Venis, Italy, pp. 533–536 (1991)
6. Bortolan, G., Brohet, C., Fusaro, S.: Possibilities of using neural networks for ECG classification. J. Electrocardiol. 29, 10–16 (1996)
7. Bosnjak, A., Bevilacqua, G., Passariello, G., Mora, F., Sanso, B., Carrault, G.: An Approach to Intelligent Ischemia Monitoring. Medical & Biomedical Engineering &Computing 33(6), 749–756 (1995)
8. Bousseljot, R.D., Kreiseler, D.: ECG analysis by signal pattern comparison. Biomedical Engineering 43, 156–157 (1998)

9. Bozzola, P., et al.: Ahybrid neuro-fuzzy system for ECG classification of myocardial infarction. In: Comput. Cardiol., Indianapolis, IN (1996)
10. Cheng, F., Gao, R., Wei, Y.: Tracing algorithms of ECG tracks from scanned image. Journal of Biomed. Eng. 18, 306–308 (2001)
11. Cheng, Q.L., Lee, H.S., Thakor, N.V.: ECG waveform analysis by significant point extraction. II. Pattern matching. Comput. Biomed. Res. 20(5), 428–442 (1987)
12. Coast, D.A., Stem, R.M., Cano, G.G., Briller, S.A.: An approach to cardiac arrhythmia analysis using hidden Markov models. IEEE Trans. Biomed. Eng. BME-37, 826–836 (1990)
13. Cromwell, L., Weibell, F., Pfeiffer, E.: Biomedical Instrumentation and Measurement, 2nd edn., pp. 384–386. Pearson Education, London (2007)
14. Degani, R.: Computerized Electrocardiogram Diagnosis: Fuzzy Approach. Methods of Inform. Med. 31, 225–233 (1992)
15. Dokur, Z., Olmez, T.: ECG beat classification by a novel hybrid neural network. Computer Methods and Programs in Biomedicine 66(2-3), 167–181 (2001)
16. Einthoven, W., et al.: Amer. Heart J. 40, 163 (1950)
17. Frank, E.: Circ. Research. 3, 243 (1955)
18. Friensen, G.M., Jannett, T.C., Jadallah, M.A., Yates, S.L., Quint, S.R., Nagle, H.T.: A comparison of the noise sensitivity of nine QRS detection algorithms. IEEE Trans. Biomed. Eng. BME-37, 85–89 (1990)
19. Kókai, G., Csirik, J., Gyimóthy, T.: Learning the syntax and semantic rules of an ECG grammar. In: Proceedings of fifth congress of the Italian Association for Artificial Intelligence, Roma (September 1997)
20. Kókai, G., Alexin, Z., Gyimóthy, T.: Learning Biomedical Patterns
21. Geddes, L.A., Baker, L.E.: Principles of applied biomedical instrumentation. John Wiley & Sons, Inc., NY (1968)
22. Gonzalez, R.C., Woods, R.E.: Digital Image Processing, 3rd edn., pp. 491–495. Addison Wesley Longman, Inc., Amsterdam (2000)
23. Goswami, B., Mitra, T.K., Mitra, M., Nag, B., Basu, S.K., Basu, D.K.: Data Base Generation from ECG Records using AutoCAD Application Package. IETE technical review 11, 67–69 (1994)
24. Goldman, M.J.: Principles of Electrocardiography, 11th edn. Marugen Asia (Pvt.) Ltd.
25. Ham, F.M., Han, S.: Classification of cardiac arrhythmias using fuzzy ARTMAP. IEEE Trans. Biomed. Eng. 43(4), 425–430 (1996)
26. Hampton, J.R.: The ECG Made Easy, 5th edn. Churchill Livingstone
27. Hildebrand, F.B.: Introduction To Numerical Analysis, T M H (edn.), pp. 82–84. Tata Mcgraw-Hill Publishing Company Ltd., New York
28. Jager, F., Mark, R.G., Moody, G.B., Divjak, S.: Analysis of transient ST segment changes during ambulatory monitoring using the Karhunen-Loeve transform. Computers in Cardiology, 691–694 (1992)
29. Ji, Z., Qin, S., Peng, C.: Electrocardiographic signal feature extraction and its instrument development based on continuous wavelet transform. Journal of Biomed. Eng. 23(6), 1186–1190 (2006)
30. Kadamb, S., Murray, R., Boudreaux-Bartels, G.F.: Wavelet transform-based QRS complex detector. IEEE Trans. Biomed. Eng. 46, 838–848 (1999)
31. King, M.J., HanK, J.S., Park, H., et al.: Classification of Arrhythmia based on discrete wavelet transform and roughset theory. In: Intl. Conf. Control, Automation and System, ICCAS (2001)

32. Koski, M., Juhola, M.: Meriste, Syntactic recognition of ECG signals by attributes finite Automata. Pattern Recognition 28(12) (1947-1940)
33. Kotsas, P., Pappas, C., Strintzis, M., Maglaveras, N.: Nonstationary ECG analysis using Wigner-Ville transform and wavelets. Computers in Cardiology, 499–502 (1993)
34. Lagerholm, M., Peterson, C.: Clustering ECG Complexes Using Hermite Functions and Self organizing Maps. IEEE Trans. Biomed. Eng. 47(7), 838–848 (2000)
35. Laguna, P., Jane, R., Caminal, P.: Automatic detection of wave boundaries in multilead ECG signals: validation with the CSE database. Comput. Biomed. Res. 27, 45–60 (1994)
36. Lee, H.S., Cheng, Q.L., Thakor, N.V.: ECG waveform analysis by significant point extraction in data reduction. Comput. Biomed. Res. 20(5), 410–427 (1987)
37. Leung, J.M., Voskanian, A., Bellows, W.H., Pastor, D.: Automated electrocardiograph ST segment trending monitors: accuracy in detecting myocardial ischemia. Anesthesia and Analgesia 87(1), 4–10 (1998)
38. Li, C., Zheng, C., Tai, C.: Detection of ECG characteristic points using wavelet transforms. IEEE Trans. Biomed. Eng. 42(1), 21–28 (1995)
39. Maglaveras, N., Stamkopoulos, T., Diamantaras, K., Pappas, C., Strintzis, M.: ECG pattern recognition and classification using non-linear transformations and neural networks: a review. International Journal of Medical Informatics 82, 191–208 (1998)
40. Maglaveras, N., Stamkopoulos, T., Pappas, C., Strintzis, M.: An adaptive backpropagation neural network for real-time ischemia episodes detection. Development and performance analysis using the European ST-T database. IEEE Trans. Biomed. Eng. 45(7), 805–813 (1998)
41. Mahmoodabadi, S., Ahmadian, A., Bolhasani, A.M., Eslami, M., Bidgoli, J.: ECG Feature Extraction Based on Multiresolution Wavelet Transform. In: Conf. Proc. IEEE Eng. Med. Biol. Soc., vol. 4, pp. 3902–3905 (2005)
42. Matsuyama, A., Jonkman, M., de Boer, F.: Improved ECG signal analysis using wavelet and feature extraction. Methods Inf. Med. 46(2), 227–230 (2007)
43. Mitra, S., Mitra, M.: An Automated Data Extraction System From 12 Lead ECG Images. Computer Methods and Programs in Biomedicine 71(1), 33–38 (2003)
44. Nugent, C.D., Webb, J.A., Black, N.D.: Feature and classifier fusion for 12-lead ECG classification. Med. Inform. Internet Med. 25(3), 225–235 (2000)
45. ϕhrn, A.: Discernibility and Rough Sets in Medicine: Tools and Applications, Ph.D. Thesis, Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, NTNU Report, 1999:133, IDI Report (1999)
46. Papaloukas, C., Fotiadis, D.I., Liavas, A.P., Likas, A., Michalis, L.K.: A knowledge-based technique for automated detection of ischaemic episodes in long duration electrocardiograms. Medical & Biological Engineering & Computing 39(1), 105–112 (2001)
47. Pavlidis, T.: Structural Pattern Recognition, pp. 185–215. Springer, Heidelberg (1980)
48. Pawlak, Z.: Bayes' Theorem Revised – The Rough Set View, New Frontiers in Artificial Intelligence: Joint. In: Terano, T., Nishida, T., Namatame, A., Tsumoto, S., Ohsawa, Y., Washio, T. (eds.) JSAI-WS 2001. LNCS, vol. 2253, pp. 240–250. Springer, Heidelberg (2001)
49. Pawlak, Z.: Rough sets. International Journal of Information and Computer Science 11(5), 341–356 (1982)

50. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning about Data. Series D: System Theory, Knowledge Engineering and Problem Solving, vol. 9. Kluwer Academic Publishers, Dordrecht (1991)
51. Pawlak, Z.: The Rough Set View on Bayes' Theorem, Advances in Soft Computing. In: Pal, N.R., Sugeno, M. (eds.) AFSS 2002. LNCS, vol. 2275, pp. 106–116. Springer, Heidelberg (2002)
52. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Information Sciences 177(1), 3–27 (2007)
53. Pawlak, Z., Skowron, A.: Rough sets: Some extensions. Information Sciences 177(1), 28–40 (2007)
54. Pawlak, Z., Skowron, A.: Rough sets and boolean reasoning. Information Sciences 177(1), 41–73 (2007)
55. Plonsey, R.: Bioelectric Phenomena (Book). Mcgraw-Hill Book company, NY (1969)
56. Polak, M.J., Zhou, S.H., Rautaharju, P.M., Armstrong, W.W., Chaitman, B.R.: Using automated analysis of the resting twealve-lead ECG to identify patients at risk of developing transient myocardial ischemia—an application of an adaptive logic network. Physiological Measurement 18(4), 317–325 (1997)
57. Polkowski, L., Skowron, A.: Rough Sets in Knowledge Discovery. Physica – Verlag, Wurzburg (1998)
58. Presedo, J., Vila, J., Barro, S., Palacios, F., Ruiz, R., Teddei, A., Emdin, M.: Fuzzy modeling of the expert's knowledge in ECG-based ischemia detection. Fuzzy Sets and Systems 77(1), 63–75 (1996)
59. Quinlan, J.R.: Induction of decision trees. MachineLearning 1, 81–106 (1986)
60. Quinlan, J.R.: C4.5 Programs for machine learning. Morgan Kaufmann, San Francisco (1993)
61. Rey, W., Laird, J.D., Hugenholtz, P.G.: P-wave detection by digital computer. Comput. Biomed. Res. 4, 509–522 (1971)
62. Scheler, A.M., et al.: Circ. Res. VIII, 519–526 (1960)
63. Sekiguchi, K., Kanda, T., Osada, M., Tsunoda, Y., Kodajima, N., Fukumura, Y., Suzuki, T., Kobayashi, I.: Comparative accuracy of automated computer analysis versus physicans in training in the interpretation of electrocardiograms. Journal of Medicine 30(1-2), 75–81 (1999)
64. Silipo, R., Laguna, P., Marchesi, C., Mark, R.G.: ST-T segment change recognition using artificialneural networks and principal component analysis. Computers in Cardiology, 213–216 (1995)
65. Skordalakis, E.: Recognition of the shape of the STsegment in ECG waveforms. IEEE Trans. Biomed. Eng. 33, 972–974 (1986)
66. Sternicke, K.: Automatic pattern recognition in ECG time series. Comput Methods Programs Biomed. 68(2), 109–115 (2002)
67. Strintzis, M., Magnisalis, X., Stalidis, G., Maglaveras, N.: Use of neural networks for electrocardiogram (ECG) feature extraction recognition and classification. Neural Network World J. 3(4), 313–327 (1992)
68. Tezuka, A.: A study on transformation system of wave data recorded on paper into digital time data. Rec. Electr. Commun. Eng. Conversazione 57, 343–344 (1989)
69. Wideman, L.E., Freeman, G.L.: A to D conversion from paper records with a desktop scanner and a microcomputer. Comp. Biomed. Res. 22, 393 (1989)
70. Wilson, F.N., et al.: Amer. Heart J. 9, 447 (1934)

71. Xiao, Y., Chen, H., Ge, J.: Automated Analysis Technology of Electrocardiogram. Journal of Biomedical Engineering 17(3), 339–342 (2000)
72. Xie, G., Nie, Z., Xiang, H., Zeng, Z.: Detection of P wave through wavelet transform and neural network. Journal of Biomed. Eng. 16(3), 320–323 (1999)
73. Item Xu, X., Liu, Y.: ECG QRS Complex Detection Using Slope Vector Waveform (SVW) Algorithm. In: Conf. Proc. IEEE Eng. Med. Biol. Soc., vol. 5, pp. 3597–3600 (2004)
74. Yu, X., Xu, X.: QRS detection based on neural-network. Journal of Biomed. Eng. 17, 59–62 (2000)

Universal Problem of Attribute Reduction

Mikhail Ju. Moshkov[1], Marcin Piliszczuk[2], and Beata Zielosko[3]

[1] Institute of Computer Science, University of Silesia
39, Będzińska St., Sosnowiec, 41-200, Poland
moshkov@us.edu.pl

[2] ING Bank Śląski S.A., 34, Sokolska St., Katowice, 40-086, Poland
marcin.piliszczuk@ingbank.pl

[3] Institute of Computer Science, University of Silesia
39, Będzińska St., Sosnowiec, 41-200, Poland
zielosko@us.edu.pl

Abstract. In the paper, some generalizations of the notions of reduct, test (superreduct), partial (approximate) reduct and partial test are considered. The accuracy of greedy algorithm for construction of partial test is investigated. A lower bound on the minimal cardinality of partial reducts based on an information about greedy algorithm work is studied. A bound on the precision of greedy algorithm which does not depend on the number of pairs of rows of a decision table which should be separated is obtained. Results of experiments with greedy algorithm are discussed.

Keywords: Partial test, partial reduct, greedy algorithm.

1 Introduction

The attribute reduction problem (it is required to find a reduct with minimal or close to minimal cardinality) is one of the main problems of rough set theory [14,16,21]. There are different variants of the notion of reduct: reducts for information systems [14], usual decision and local reducts for decision tables [14,20], decision and local reducts which are based on the generalized decision [20], etc. Interesting discussion of various kinds of reducts can be found in [16], page 12.

In this paper, we consider "universal" definition of reduct which covers at least part of possible variants. We use an approach considered in test theory [30]. Let T be a decision table and $\mathcal{P}$ be a subset of pairs of discernible rows (objects) of T. Then a reduct for T relative to $\mathcal{P}$ is a minimal (relative to inclusion) subset of conditional attributes which separate all pairs from $\mathcal{P}$. All mentioned above kinds of reducts can be represented in such a form.

In this paper, we consider not only exact but also approximate (partial) reducts, which are useful in inducing data models. Rough set theory often deals with information and decision systems containing noisy data. In this case, the exact reducts can be "over-learned", i.e., depend essentially on the noise. If we view reducts as a way of knowledge representation [20], then instead of large exact reducts it is more appropriate to work with relatively smaller, inexact ones. The approximate reducts have been studied very intensively since the 1990s [6,7,9,11,12,17,18,24,25,27,28,29,31,32]. They became especially popular area of research beginning from 1998, according to the following words of Zdzisław Pawlak: "the idea of an approximate reduct can be useful in cases when a smaller number of condition attributes is preferred over accuracy of classification" [15].

We begin our consideration from a data table, which columns are labeled by discrete and continuous variables, and rows are tuples of values of variables on some objects. It is possible that this data table contains missing values [2,4]. We consider the following classification problem: for a discrete variable we should find its value using values of all other variables. We do not use variables directly but create some attributes with relatively small number of values based on the considered variables. As a result, we obtain a decision table with missing values in the general case. We define the universal attribute reduction problem for this table and consider a number of examples of known attribute reduction problems which can be represented as the universal one.

Based on results from [9], we obtain bounds on precision of greedy algorithm for partial test (superreduct) construction. This algorithm is a simple generalization of greedy algorithm for set cover problem [3,5,13,22,23]. We prove that under some natural assumptions on the class NP the greedy algorithm is close to the best (from the point of view of precision) polynomial approximate algorithms for minimization of cardinality of partial tests. We show that based on an information received during greedy algorithm work it is possible to obtain a nontrivial lower bound on minimal cardinality of partial reduct. We obtain also a bound on precision of greedy algorithm which does not depend on the cardinality of the set $\mathcal{P}$.

In [6,7,9] we described results of experiments with randomly generated decision tables. In this paper we discuss results of experiments with real-life decision tables from [10]. Part of these results illustrates the use of lower bound on minimal cardinality of partial reducts based on an information received during greedy algorithm work (see Theorem 7). This bound can be useful in experiments connected with the construction of various kinds of reducts by greedy algorithm.

This paper is an extended version of [8].

The paper consists of five sections. In Sect. 2 a transformation of a data table into a decision table is considered. In Sect. 3 the notion of the universal attribute reduction problem is discussed. In Sect. 4 greedy algorithm for construction of partial tests (partial superreducts) is studied. In Sect. 5 results of experiments with some decision tables from [10] are discussed.

2 From Data Table to Decision Table

The data table D is a finite table with k columns labeled by variables $x_1, \ldots, x_k$ and N rows which are interpreted as tuples of values of variables $x_1, \ldots, x_k$ on N objects $u_1, \ldots, u_N$. It is possible that D contains missing values which are denoted by " $-$ ".

As usual, we assume that each of variables x_i is either discrete (with values from some finite unordered set $V(x_i)$) or continuous (with values from a set $V(x_i) \subset \mathbb{R}$). We will assume that " $-$ " does not belong to $V(x_i)$.

Let us choose a variable $x_r \in \{x_1, \ldots, x_k\}$ and consider the problem of prediction of the value of x_r on a given object using only values of variables from the set $X = \{x_1, \ldots, x_k\} \setminus \{x_r\}$ on the considered object. If x_r is a discrete variable, then the problem of prediction is called the classification problem. If x_r is a continuous variable, then the considered problem is called the problem of regression. We only consider the classification problem. So x_r is a discrete variable.

We only consider two kinds of missing values: (i) missing value of x_i as an additional value of variable x_i, which does not belong to $V(x_i)$, and (ii) missing value as an undefined value. In the last case, based on the value of x_i it is impossible to discern an object u_l from another object u_t if the value $x_i(u_l)$ is missing (undefined).

We now transform the data table D into a data table D^*. For each variable $x_i \in \{x_1, \ldots, x_k\}$, according to the nature of x_i we choose either the first or the second way for the work with missing values. In the first case, we add to $V(x_i)$ a new value which is not equal to " $-$ ", and write this new value instead of each missing value of x_i. In the second case, we leave all missing values of x_i untouched.

To solve the considered classification problem, we do not use variables from X directly. Instead of this we use attributes constructed on the basis of these variables. Let us consider some examples.

Let $x_i \in X$ be a discrete variable. Let us divide the set $V(x_i)$ into relatively small number of nonempty disjoint subsets $V_1, \ldots, V_s$. Then the value of the considered attribute on an object u is equal to the value $j \in \{1, \ldots, s\}$ for which $x_i(u) \in V_j$. The value of this attribute on u is missing if and only if the value of x_i on u is missing.

Let $x_i \in X$ be a continuous variable and $c \in \mathbb{R}$. Then the value of the considered attribute on an object u is equal to 0 if $x_i(u) < c$, and is equal to 1 otherwise. The value of this attribute on u is missing if and only if the value of x_i on u is missing.

Let $x_{i_1}, \ldots, x_{i_t} \in X$ be continuous variables and f be a function from $\mathbb{R}^t$ to $\mathbb{R}$. Then the value of the considered attribute on an object u is equal to 0 if $f(x_{i_1}(u), \ldots, x_{i_t}(u)) < 0$, and is equal to 1 otherwise. The value of this attribute on u is missing if and only if the value of at least one variable from $\{x_{i_1}, \ldots, x_{i_t}\}$ on u is missing.

We now assume that the attributes $a_1, \ldots, a_m$ are chosen. Let, for the definiteness, $u_1, \ldots, u_n$ be all objects from $\{u_1, \ldots, u_N\}$ such that the value of the variable x_r on the considered object is definite (is not missing).

We now describe a decision table T. This table contains m columns labeled by attributes $a_1, \ldots, a_m$, and n rows corresponding to objects $u_1, \ldots, u_n$ respectively. For $j = 1, \ldots, n$ the j-th row is labeled by the value $x_r(u_j)$, which will be considered later as the value of the decision attribute d. For any $i \in \{1, \ldots, m\}$

and $j \in \{1, \ldots, n\}$ the value $a_i(u_j)$ is at the intersection of the j-th row and the i-th column. If the value $a_i(u_j)$ is missing then the symbol " $-$ " is at the intersection of the j-th row and the i-th column.

3 Problem of Attribute Reduction

3.1 Definition of Problem

Let T be a decision table with m columns labeled by attributes $a_1, \ldots, a_m$ and n rows which are identified with objects $u_1, \ldots, u_n$. It is possible that T contains missing values denoted by " $-$ ". Each row is labeled by a decision which is interpreted as the value of the decision attribute d. Let $A = \{a_1, \ldots, a_m\}$ and $U = \{u_1, \ldots, u_n\}$.

We now define the indiscernibility relation $IND(T) \subseteq U \times U$. Let $u_l, u_t \in U$. Then $(u_l, u_t) \in IND(T)$ if and only if $a_i(u_l) = a_i(u_t)$ for any $a_i \in A$ such that the values $a_i(u_l)$ and $a_i(u_t)$ are definite (are not missing). Since T can contain missing values, the relation $IND(T)$ is not an equivalence relation in the general case, but it is a tolerance relation.

By $DIS(T)$ we denote the set of unordered pairs of objects u_l and u_t from U such that $(u_l, u_t) \notin IND(T)$. Let $(u_l, u_t) \in DIS(T)$ and $a_i \in A$. We will say that the attribute a_i separates the pair (u_l, u_t) if the values $a_i(u_l)$ and $a_i(u_t)$ are definite and $a_i(u_l) \neq a_i(u_t)$. For any $a_i \in A$ we denote by $DIS(T, a_i)$ the set of pairs from $DIS(T)$ which the attribute a_i separates.

Let $\mathcal{P}$ be a subset of $DIS(T)$. Let Q be a subset of A and α be a real number such that $0 \leq \alpha < 1$. We will say that Q is an α-test for T relative to $\mathcal{P}$ (an $(\alpha, \mathcal{P})$-test for T) if attributes from Q separate at least $(1-\alpha)|\mathcal{P}|$ pairs from $\mathcal{P}$. An $(\alpha, \mathcal{P})$-test for T is called an α-reduct for T relative to $\mathcal{P}$ (an $(\alpha, \mathcal{P})$-reduct for T) if each proper subset of this $(\alpha, \mathcal{P})$-test is not an $(\alpha, \mathcal{P})$-test for T. If $\mathcal{P} = \emptyset$, then any subset Q of A is an $(\alpha, \mathcal{P})$-test for T, but only the empty set of attributes is an $(\alpha, \mathcal{P})$-reduct for T. Note that each $(\alpha, \mathcal{P})$-test contains an $(\alpha, \mathcal{P})$-reduct as a subset. The parameter α can be interpreted as inaccuracy. If $\alpha = 0$, then we obtain the notion of exact test for T relative to $\mathcal{P}$ and the notion of exact reduct for T relative to $\mathcal{P}$.

The problem of attribute reduction is the following: for a given decision table T, subset $\mathcal{P}$ of the set $DIS(T)$ and real α, $0 \leq \alpha < 1$, it is required to find an $(\alpha, \mathcal{P})$-reduct for T (an $(\alpha, \mathcal{P})$-test for T) with minimal cardinality. Let us denote by $R_{\min}(\alpha) = R_{\min}(\alpha, \mathcal{P}, T)$ the minimal cardinality of an $(\alpha, \mathcal{P})$-reduct for T. Of course, it is possible to use another measures of reduct quality.

The considered problem can be easily reformulated as a set cover problem: we should cover the set $\mathcal{P}$ using minimal number of subsets from the family $\{\mathcal{P} \cap DIS(T, a_1), \ldots, \mathcal{P} \cap DIS(T, a_m)\}$. Therefore, we can use results, obtained for the set cover problem, for analysis of the attribute reduction problem.

3.2 Examples

Now we consider examples of sets $\mathcal{P}$ corresponding to different kinds of reducts. It was impossible for us to find definitions of some kinds of reducts which are applicable to decision tables with missing values. In such cases we have extended existing definitions (if it was possible) trying to preserve their spirit.

For an arbitrary $u_l \in U$, let $[u_l]_T = \{u_t : u_t \in U, (u_l, u_t) \in IND(T)\}$ and $\partial_T(u_l) = \{d(u_t) : u_t \in [u_l]_T\}$. The set $\partial_T(u_l)$ is called the generalized decision for u_l. The positive region $POS(T)$ for T is the set of objects $u_l \in U$ such that $|\partial_T(u_l)| = 1$. The set $BN(T) = U \setminus POS(T)$ is called the boundary region for T.

1. *Reducts for the information system, obtained from T by removing the decision attribute d.* The set $\mathcal{P}$ is equal to $DIS(T)$ (we must preserve the indiscernibility relation).
2. *Usual decision reducts for T.* The set $\mathcal{P}$ is equal to the set of all pairs $(u_l, u_t) \in DIS(T)$ such that $d(u_l) \neq d(u_t)$ and at least one object from the pair belongs to $POS(T)$ (we must preserve the positive region).
3. *Decision reducts for T based on the generalized decision.* Let us assume T is without missing values. The set $\mathcal{P}$ is equal to the set of all pairs $(u_l, u_t) \in DIS(T)$ such that $\partial_T(u_l) \neq \partial_T(u_t)$.
4. *Maximally discerning decision reducts for T.* The set $\mathcal{P}$ is equal to the set of all pairs $(u_l, u_t) \in DIS(T)$ such that $d(u_l) \neq d(u_t)$.
5. *Usual local reducts for T and object $u_l \in POS(T)$.* The set $\mathcal{P}$ is equal to the set of all pairs $(u_l, u_t) \in DIS(T)$ such that $d(u_l) \neq d(u_t)$.
6. *Local reducts for T and object $u_l \in U$ based on the generalized decision.* Let us assume T is without missing values. The set $\mathcal{P}$ is equal to the set of all pairs $(u_l, u_t) \in DIS(T)$ such that $\partial_T(u_l) \neq \partial_T(u_t)$.
7. *Maximally discerning local reducts for T and object $u_l \in U$.* The set $\mathcal{P}$ is equal to the set of all pairs $(u_l, u_t) \in DIS(T)$ such that $d(u_l) \neq d(u_t)$.

3.3 Maximally Discerning Reducts

The notions of maximally discerning decision and local reducts (but without the use of the term "maximally discerning") were investigated by the authors in [6,7,9,17,32]. Note also that similar notions were considered earlier in [26,27]. Maximally discerning decision reducts can give us additional information on the value of the decision attribute (for example, by the separation of groups of equal rows with the same generalized decision but with different probability distributions of decision values). The consideration of maximally discerning local reducts for objects from the boundary region can lead to construction of a decision rule system which is applicable to wider class of new objects. We now consider two examples.

Example 1. Let us consider the decision table T_1 (see Figure 1). For this table, there is exactly one usual decision reduct (which is equal to the empty set), exactly one decision reduct based on the generalized decision (which is equal to the empty set too) and exactly one maximally discerning decision reduct (which is equal to $\{a_2\}$). Based on reducts of the first two kinds it is impossible to separate the rows $(0, 0)$ from the rows $(0, 1)$. However, for the considered two types of rows we have different probability distributions of decision values. The third kind of reducts allows us to separate these two types of rows.

T_1

a_1	a_2	
0	0	1
0	0	2
0	1	1
0	1	2
0	1	2

T_2

a_1	a_2	
0	0	1
0	0	2
0	1	2
1	0	1

S_1

$a_2 = 1 \rightarrow 2$
$a_1 = 1 \rightarrow 1$

S_2

$a_1 = 0 \wedge a_2 = 0 \rightarrow \{1, 2\}$
$a_2 = 1 \rightarrow \{2\}$
$a_1 = 1 \rightarrow \{1\}$

S_3

$a_2 = 0 \rightarrow 1$
$a_1 = 0 \rightarrow 2$
$a_2 = 1 \rightarrow 2$
$a_1 = 1 \rightarrow 1$

Fig. 1. Illustrations to Examples 1 and 2

Example 2. Let us consider the decision table T_2 and three systems of decision rules S_1, S_2 and S_3 obtained on the basis of usual local reducts, local reducts based on the generalized decision and maximally discerning local reducts (see Figure 1). Let us consider two new objects $(0, 2)$ and $(2, 0)$. Systems S_1 and S_2 have no rules which are realizable on the new objects. However, the system S_3 has rules which are realizable on these new objects and moreover, correspond to these objects different decisions.

4 Greedy Algorithm

Now we describe the greedy algorithm which for given α, $0 \leq \alpha < 1$, decision table T and set of pairs $\mathcal{P} \subseteq DIS(T)$, $\mathcal{P} \neq \emptyset$, constructs an $(\alpha, \mathcal{P})$-test for T.

Algorithm 1. Greedy algorithm for partial test construction.

```
Input  : Decision table $T$ with conditional attributes $a_1, \ldots, a_m$, set of pairs
         $\mathcal{P} \subseteq DIS(T)$, $\mathcal{P} \neq \emptyset$, and real number $\alpha$, $0 \leq \alpha < 1$.
Output: $(\alpha, \mathcal{P})$-test for $T$.
$Q \longleftarrow \emptyset$;
while $Q$ is not an $(\alpha, \mathcal{P})$-test for $T$ do
    select $a_i \in \{a_1, \ldots, a_m\}$ with minimal index $i$ such that $a_i$ separates the
    maximal number of pairs from $\mathcal{P}$ unseparated by attributes from $Q$;
    $Q \longleftarrow Q \cup \{a_i\}$;
end
return $Q$;
```

By $R_{\text{greedy}}(\alpha) = R_{\text{greedy}}(\alpha, \mathcal{P}, T)$ we denote the cardinality of the constructed $(\alpha, \mathcal{P})$-test for T.

4.1 On Precision of Greedy Algorithm

The following three theorems are simple corollaries of results from [22,23,9].

Theorem 1. *Let* $0 \leq \alpha < 1$ *and* $\lceil (1 - \alpha)|\mathcal{P}| \rceil \geq 2$. *Then*

$$R_{\text{greedy}}(\alpha) < R_{\min}(\alpha) \cdot (\ln \lceil (1 - \alpha)|\mathcal{P}| \rceil - \ln \ln \lceil (1 - \alpha)|\mathcal{P}| \rceil + 0.78) .$$

Theorem 2. *Let $0 \le \alpha < 1$. Then for any natural $t \ge 2$ there exists a decision table T and a subset $\mathcal{P}$ of the set $DIS(T)$ such that $\lceil (1-\alpha)|\mathcal{P}| \rceil = t$ and*

$$R_{\text{greedy}}(\alpha) > R_{\min}(\alpha)(\ln \lceil (1-\alpha)|\mathcal{P}| \rceil - \ln \ln \lceil (1-\alpha)|\mathcal{P}| \rceil - 0.31) \ .$$

Theorem 3. *Let $0 \le \alpha < 1$. Then*

$$R_{\text{greedy}}(\alpha) \le R_{\min}(\alpha)(1 + \ln(\max_{j \in \{1,\ldots,m\}} |\mathcal{P} \cap DIS(T, a_j)|)) \ .$$

4.2 On Polynomial Approximate Algorithms

Immediately from results obtained in [12,28] the next theorem follows.

Theorem 4. *Let $0 \le \alpha < 1$. Then the problem of construction, for given T and $\mathcal{P} \subseteq DIS(T)$, an $(\alpha, \mathcal{P})$-reduct for T with minimal cardinality is NP-hard.*

From statements obtained in [9] (based on results from [1,19,25,28]) the next two theorems follow.

Theorem 5. *Let $\alpha \in \mathbb{R}$ and $0 \le \alpha < 1$. If $NP \not\subseteq DTIME(n^{O(\log \log n)})$, then for any ε, $0 < \varepsilon < 1$, there is no polynomial algorithm that, for given decision table T with $DIS(T) \ne \emptyset$ and nonempty subset $\mathcal{P} \subseteq DIS(T)$, constructs an $(\alpha, \mathcal{P})$-test for T which cardinality is at most $(1-\varepsilon) R_{\min}(\alpha, \mathcal{P}, T) \ln |\mathcal{P}|$.*

From Theorem 3 it follows that $R_{\text{greedy}}(\alpha) \le R_{\min}(\alpha)(1 + \ln |\mathcal{P}|)$. From this inequality and from Theorem 5 it follows that under the assumption $NP \not\subseteq DTIME(n^{O(\log \log n)})$ the greedy algorithm is close to the best polynomial approximate algorithms for partial test cardinality minimization.

Theorem 6. *Let α be a real number such that $0 \le \alpha < 1$. If $P \ne NP$, then there exists $\rho > 0$ such that there is no polynomial algorithm that, for given decision table T with $DIS(T) \ne \emptyset$ and nonempty subset $\mathcal{P} \subseteq DIS(T)$, constructs an $(\alpha, \mathcal{P})$-test for T which cardinality is at most $\rho R_{\min}(\alpha, \mathcal{P}, T) \ln |\mathcal{P}|$.*

From Theorems 3 and 6 it follows that under the assumption $P \ne NP$ the greedy algorithm is not far from the best polynomial approximate algorithms for partial test cardinality minimization.

4.3 Lower Bound on $R_{\min}(\alpha)$

In this subsection, we fix some information about greedy algorithm work and find a lower bound on $R_{\min}(\alpha)$ depending on this information.

Let us apply the greedy algorithm to α, T and $\mathcal{P}$. Let during the construction of $(\alpha, \mathcal{P})$-test for T the greedy algorithm choose consequently attributes $a_{j_1}, \ldots, a_{j_t}$. Let us denote by δ_1 the number of pairs from $\mathcal{P}$ separated by the attribute a_{j_1}. For $i = 2, \ldots, t$ we denote by δ_i the number of pairs from $\mathcal{P}$ which are not separated by attributes $a_{j_1}, \ldots, a_{j_{i-1}}$ but are separated by the attribute

a_{j_i}. Let $\Delta(\alpha, \mathcal{P}, T) = (\delta_1, \ldots, \delta_t)$. As information on the greedy algorithm work we will use the tuple $\Delta(\alpha, \mathcal{P}, T)$ and numbers $|\mathcal{P}|$ and α.

We now define the parameter $l(\alpha) = l(\alpha, |\mathcal{P}|, \Delta(\alpha, \mathcal{P}, T))$. Let $\delta_0 = 0$. Then

$$l(\alpha) = \max \left\{ \left\lceil \frac{\lceil (1-\alpha)|\mathcal{P}| \rceil - (\delta_0 + \ldots + \delta_i)}{\delta_{i+1}} \right\rceil : i = 0, \ldots, t-1 \right\} .$$

Next two theorems follow immediately from results obtained in [9].

Theorem 7. *Let T be a decision table, $\mathcal{P} \subseteq DIS(T)$, $\mathcal{P} \neq \emptyset$, and α be a real number such that $0 \le \alpha < 1$. Then*

$$R_{\min}(\alpha, \mathcal{P}, T) \ge l(\alpha, |\mathcal{P}|, \Delta(\alpha, \mathcal{P}, T)) .$$

The value $l(\alpha) = l(\alpha, |\mathcal{P}|, \Delta(\alpha, \mathcal{P}, T))$ can be used for the obtaining upper bounds on cardinality of partial tests constructed by the greedy algorithm.

Theorem 8. *Let α and β be real numbers such that $0 < \beta \le \alpha < 1$. Then*

$$R_{\text{greedy}}(\alpha) < l(\alpha - \beta) \ln \left(\frac{1 - \alpha + \beta}{\beta} \right) + 1 .$$

From Theorem 8 it follows that the lower bound $R_{\min}(\alpha) \ge l(\alpha)$ is nontrivial. In [9] it is shown that for maximally discerning decision reducts and maximally discerning local reducts the bound $R_{\min}(\alpha) \ge l(\alpha)$ is the best lower bound for $R_{\min}(\alpha)$ depending on $\Delta(\alpha, \mathcal{P}, T)$, $|\mathcal{P}|$ and α.

4.4 Upper Bound on $R_{\text{greedy}}(\alpha)$

In this subsection, we obtain an upper bound on $R_{\text{greedy}}(\alpha) = R_{\text{greedy}}(\alpha, \mathcal{P}, T)$ which does not depend on $|\mathcal{P}|$. The next statement follows immediately from Theorems 7 and 8.

Theorem 9. *Let α and β be real numbers such that $0 < \beta \le \alpha < 1$. Then*

$$R_{\text{greedy}}(\alpha) < R_{\min}(\alpha - \beta) \ln \left(\frac{1 - \alpha + \beta}{\beta} \right) + 1 .$$

In [9] it is shown that for maximally discerning decision reducts and maximally discerning local reducts this bound is, in some sense, unimprovable: it is impossible to multiply the right hand side of the considered inequality by any real δ such that $\delta < 1$.

5 Results of Experiments

Results of experiments with randomly generated decision tables and some theoretical results from [9] give us arguments in behalf of the following hypothesis: for almost all decision tables during each step the greedy algorithm (under the construction of maximally discerning decision test) chooses an attribute which separates at least one half of unseparated pairs of rows. It is not difficult to show that in such cases $R_{\text{greedy}}(\alpha) \le \lceil \log_2 \frac{1}{\alpha} \rceil$ for $\alpha > 0$, and $l(\alpha) \le 2$ for any α. In particular, $R_{\text{greedy}}(0.1) \le 4$, $R_{\text{greedy}}(0.01) \le 7$, and $R_{\text{greedy}}(0.001) \le 10$. So using greedy algorithm it is possible to construct for such tables short partial tests with relatively high accuracy.

To understand the situation with real-life decision tables we made additional experiments with about 20 tables from [10]. For each of the considered tables T in the capacity of the set $\mathcal{P}$ we choose the set of all pairs $(u_l, u_t) \in DIS(T)$ such that $d(u_l) \neq d(u_t)$ (so we study maximally discerning decision reducts). We apply to the decision table T, the set $\mathcal{P}$ and $\alpha = 0$ the greedy algorithm.

The main result of these experiments is the following: with the exception of the tables "kr-vs-kp" and "spect" during each step the greedy algorithm chooses an attribute which separates at least one half of unseparated pairs. It means that not only for randomly generated tables but also for real-life tables it is possible to construct short partial tests with relatively high accuracy using greedy algorithm.

Details of experiments with decision tables "kr-vs-kp", "poker-hand-training-true", "nursery", "monks-2.train" and "soybean-small" can be found in Tables 1 - 3.

The column "#" contains the number i of step of greedy algorithm, the column "attr." contains the name of attribute chosen during the i-th step, the column "α" contains the inaccuracy of partial test constructed during the first i steps, the column "%" contains the percentage of unseparated during first $i-1$ steps pairs which are separated during the i-th step, and the column "$l(\alpha)$" (for Table 1)

Table 1. Results of the experiment with the decision table "kr-vs-kp" (36 conditional attributes and 3196 rows, $|\mathcal{P}| = 2,548,563$)

#	%	α	$l(\alpha)$	attr.
1	53.86	0.461391687916226	1.0	wknck
2	55.80	0.203919051411864	2.0	bxqsq
3	55.01	0.091742516013002	2.0	wkpos
4	59.55	0.037112982420434	2.0	rimmx
5	54.48	0.016895013806505	2.0	bkxbq
6	55.69	0.007487020319340	2.0	katri
7	49.28	0.003797718550816	2.0	simpl
8	45.27	0.002078314490862	2.0	r2ar8
9	45.12	0.001140528893855	2.0	blxwp
10	39.91	0.000685334966761	2.0	dwipd
11	37.98	0.000425056409995	2.0	bkspr
12	40.70	0.000252059233920	3.0	cntxt
13	31.37	0.000172997176076	3.0	skewr
14	21.72	0.000135423128783	3.0	rxmsq
15	23.70	0.000103328630054	3.0	wkovl
16	22.73	0.000079844850496	3.0	bkblk
17	26.47	0.000058709448894	4.0	wtoeg
18	24.00	0.000044619181160	4.0	mulch
19	24.56	0.000033660084033	4.0	thrsk
20	29.07	0.000023875175884	4.0	reskr
21	29.51	0.000016830042016	5.0	qxmsq
22	37.21	0.000010567700801	5.0	bkxcr
23	29.63	0.000007436530193	5.0	skrxp
24	31.58	0.000005088152238	5.0	bkona
25	38.46	0.000003131170608	5.0	skach
26	37.50	0.000001956981630	5.0	wkcti
27	40.00	0.000001174188978	5.0	bkon8
28	66.67	0.000000391396326	5.0	dsopp
29	100.0	0.000000000000000	5.0	spcop

Table 2. Results of experiments with the decision table "poker-hand-training-true" (10 conditional attributes and 25010 rows, $|\mathcal{P}| = 77,677,849$) and the decision table "nursery" (8 conditional attributes and 12960 rows, $|\mathcal{P}| = 57,319,460$)

"poker-hand-training-true"

#	%	α	attr.
1	92.31	0.076908180039933	C2
2	92.57	0.005713362727618	C5
3	93.07	0.000396183319396	C4
4	94.29	0.000022608333130	C3
5	99.80	0.000000045025309	C1
6	87.50	0.000000005628164	S1
7	100.0	0.000000000000000	S2

"nursery"

#	%	α	attr.
1	83.09	0.169147720512370	health
2	86.40	0.023000984307947	has_nurs
3	77.33	0.005214703697488	parents
4	77.44	0.001176284633526	children
5	77.85	0.000260574680920	form
6	77.08	0.000059735384806	housing
7	87.03	0.000007746060413	social
8	100.0	0.000000000000000	finance

Table 3. Results of experiments with the decision table "monks-2.train" (6 conditional attributes (we ignore the attribute "Id") and 169 rows, $|\mathcal{P}| = 6720$) and the decision table "soybean-small" (35 conditional attributes and 47 rows, $|\mathcal{P}| = 810$)

"monks-2.train"

#	%	α	attr.
1	75.27	0.247321428571429	a5
2	67.99	0.079166666666667	a4
3	66.92	0.026190476190476	a1
4	67.05	0.008630952380952	a2
5	63.79	0.003125000000000	a3
6	100.0	0.000000005628164	a6

"soybean-small"

#	%	α	attr.
1	92.59	0.074074074074074	canker-lesion
2	100.0	0.000000000000000	temp

contains the lower bound on minimal cardinality of $(\alpha, \mathcal{P})$-test. We do not study the lower bound $l(\alpha)$ for tables "poker-hand-training-true", "nursery", "monks-2.train" and "soybean-small" since for these tables the considered lower bound is at most 2.

For the table "kr-vs-kp" the maximal value of $l(\alpha)$ is equal to 5. For this table $R_{\text{greedy}}(0.1) = 3$, $R_{\text{greedy}}(0.01) = 6$, $R_{\text{greedy}}(0.001) = 10$, and $R_{\text{greedy}}(0) = 29$. So instead of long exact test (29 attributes) one can work with short partial tests with relatively high accuracy, for example, with a partial test containing 6 attributes which separate more than 99% pairs of rows.

For tables "poker-hand-training-true", "nursery", "monks-2.train" and "soybean-small" there exist short partial tests with relatively high accuracy: for the table "poker-hand-training-true" 4 attributes separate more than 99.99% pairs of rows, for the table "nursery" 5 attributes separate more than 99.9% pairs of rows, for the table "monks-2.train" 4 attributes separate more than 99% pairs of rows and for the table "soybean-small" 2 attributes separate 100% pairs of rows.

6 Conclusions

The paper is devoted to discussion of universal attribute reduction problem and to analysis of greedy algorithm for solving this problem. The obtained results show that, under some natural assumptions on the class NP, greedy algorithm is close to the best polynomial approximate algorithms for the minimization of partial test cardinality. Based on an information received during the greedy algorithm work it is possible to obtain lower bound on the minimal cardinality of partial reducts. Experimental results show that using greedy algorithm we often can construct short partial tests with relatively high accuracy.

Acknowledgement

The authors are greatly indebted to Andrzej Skowron for stimulating discussions, and to the anonymous referee for helpful comments.

References

1. Feige, U.: A threshold of $\ln n$ for approximating set cover (Preliminary version). In: Proceedings of 28th Annual ACM Symposium on the Theory of Computing, pp. 314–318 (1996)
2. Grzymala-Busse, J.W.: Data with missing attribute values: Generalization of indiscernibility relation and rule induction. In: Peters, J.F., Skowron, A., Grzymała-Busse, J.W., Kostek, B.z., Świniarski, R.W., Szczuka, M.S. (eds.) Transactions on Rough Sets I. LNCS, vol. 3100, pp. 78–95. Springer, Heidelberg (2004)
3. Johnson, D.S.: Approximation algorithms for combinatorial problems. J. Comput. System Sci. 9, 256–278 (1974)
4. Kryszkiewicz, M.: Rules in incomplete information systems. Information Sciences 113, 271–292 (1999)
5. Lovász, L.: On the ratio of optimal integral and fractional covers. Discrete Math. 13, 383–390 (1975)
6. Moshkov, M.J., Piliszczuk, M., Zielosko, B.: On construction of partial reducts and irreducible partial decision rules. Fundamenta Informaticae 75(1-4), 357–374 (2007)
7. Moshkov, M.J., Piliszczuk, M., Zielosko, B.: On partial covers, reducts and decision rules with weights. In: Peters, J.F., Skowron, A., Düntsch, I., Grzymała-Busse, J.W., Orłowska, E., Polkowski, L. (eds.) Transactions on Rough Sets VI. LNCS, vol. 4374, pp. 211–246. Springer, Heidelberg (2007)
8. Moshkov, M.J., Piliszczuk, M., Zielosko, B.: Universal attribute reduction problem. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS, vol. 4585, pp. 417–426. Springer, Heidelberg (2007)
9. Moshkov, M.J., Piliszczuk, M., Zielosko, B.: On partial covers, reducts and decision rules. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets VIII. LNCS, vol. 5084, pp. 258–296. Springer, Heidelberg (2008)
10. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI Repository of machine learning databases. University of California, Irvine, Department of Information and Computer Sciences (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`

11. Nguyen, H.S.: Approximate Boolean reasoning: foundations and applications in data mining. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets V. LNCS, vol. 4100, pp. 344–523. Springer, Heidelberg (2006)
12. Nguyen, H.S., Ślęzak, D.: Approximate reducts and association rules - correspondence and complexity results. In: Zhong, N., Skowron, A., Ohsuga, S. (eds.) RSFD-GrC 1999. LNCS, vol. 1711, pp. 137–145. Springer, Heidelberg (1999)
13. Nigmatullin, R.G.: Method of steepest descent in problems on cover. In: Memoirs of Symposium Problems of Precision and Efficiency of Computing Algorithms, Kiev, USSR, vol. 5, pp. 116–126 (1969) (in Russian)
14. Pawlak, Z.: Rough Sets – Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht (1991)
15. Pawlak, Z.: Rough set elements. In: Polkowski, L., Skowron, A. (eds.) Rough Sets in Knowledge Discovery 1. Methodology and Applications (Studies in Fuzziness and Soft Computing 18), pp. 10–30. Phisica-Verlag, Springer-Verlag (1998)
16. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Information Sciences 177(1), 3–27 (2007); Rough sets: Some extensions. Information Sciences 177(1), 28–40 (2007); Rough sets and boolean reasoning. Information Sciences 177(1), 41–73 (2007)
17. Piliszczuk, M.: On greedy algorithm for partial reduct construction. In: Proceedings of Concurrency, Specification and Programming Workshop, Ruciane-Nida, Poland, vol. 2, pp. 400–411 (2005)
18. Quafafou, M.: α-RST: a generalization of rough set theory. Information Sciences 124, 301–316 (2000)
19. Raz, R., Safra, S.: A sub-constant error-probability low-degree test, and sub-constant error-probability PCP characterization of NP. In: Proceedings of 29th Annual ACM Symposium on the Theory of Computing, pp. 475–484 (1997)
20. Skowron, A.: Rough sets in KDD. In: Proceedings of the 16th World Computer Congress (IFIP 2000), Beijing, China, pp. 1–14 (2000)
21. Skowron, A., Rauszer, C.: The discernibility matrices and functions in information systems. In: Slowinski, R. (ed.) Intelligent Decision Support. Handbook of Applications and Advances of the Rough Set Theory, pp. 331–362. Kluwer Academic Publishers, Dordrecht (1992)
22. Slavík, P.: A tight analysis of the greedy algorithm for set cover (extended abstract). In: Proceedings of 28th Annual ACM Symposium on the Theory of Computing, pp. 435–441 (1996)
23. Slavík, P.: Approximation algorithms for set cover and related problems. Ph.D. thesis. University of New York, Buffalo (1998)
24. Ślęzak, D.: Approximate reducts in decision tables. In: Proceedings of the Congress Information Processing and Management of Uncertainty in Knowledge-based Systems, Granada, Spain, vol. 3, pp. 1159–1164 (1996)
25. Ślęzak, D.: Normalized decision functions and measures for inconsistent decision tables analysis. Fundamenta Informaticae 44, 291–319 (2000)
26. Ślęzak, D.: Various approaches to reasoning with frequency based decision reducts: a survey. In: Polkowski, L., Tsumoto, S., Lin, T.Y. (eds.) Rough Set Methods and Applications, pp. 235–285. Physica-Verlag, Heidelberg (2000)
27. Ślęzak, D.: Approximate decision reducts. Ph.D. thesis. Warsaw University (2001) (in Polish)
28. Ślęzak, D.: Approximate entropy reducts. Fundamenta Informaticae 53, 365–390 (2002)

29. Wróblewski, J.: Ensembles of classifiers based on approximate reducts. Fundamenta Informaticae 47, 351–360 (2001)
30. Yablonskii, S.V.: Tests. In: Glushkov, V.M. (ed.) Encyclopedia of Cybernetics (in Russian), pp. 431–432. Main Editorial Board of Ukrainian Soviet Encyclopedia, Kiev (1975)
31. Ziarko, W.: Analysis of uncertain information in the framework of variable precision rough sets. Foundations of Computing and Decision Sciences 18, 381–396 (1993)
32. Zielosko, B.: On partial decision rules. In: Proceedings of Concurrency, Specification and Programming Workshop, Ruciane-Nida, Poland, vol. 2, pp. 598–609 (2005)

Extracting Relevant Information about Reduct Sets from Data Tables

Mikhail Ju. Moshkov[1], Andrzej Skowron[2], and Zbigniew Suraj[3]

[1] Institute of Computer Science, University of Silesia
Będzińska 39, 41-200 Sosnowiec, Poland
moshkov@us.edu.pl

[2] Institute of Mathematics, Warsaw University
Banacha 2, 02-097 Warsaw, Poland
skowron@mimuw.edu.pl

[3] Chair of Computer Science, University of Rzeszów
Rejtana 16A, 35-310 Rzeszów, Poland
zsuraj@univ.rzeszow.pl

Abstract. The direct searching for relevant reducts in the set of all reducts of a given data table can be often computationally infeasible, especially for large data tables. Hence, there is a need for developing efficient methods for extracting relevant information about reducts from data tables which could help us to perform efficiently the inducing process of the high quality data models such as rule based classifiers. Such relevant information could help, e.g., to reduce the dimensionality of the attribute set. We discuss methods for generating relevant information about reduct sets from information systems or decision tables. In particular, we consider a binary relation on attributes satisfied for two given attributes if and only if there is no reduct consisting them both. Moreover, we prove that for any fixed natural k, there exists a polynomial in time algorithm which for a given decision table T and given k conditional attributes recognizes if there exists a decision reduct of T covering these k attributes. We also present a list of problems related to the discussed issues. The reported results create a step toward construction of a software library reducing the searching costs for relevant reducts.

Keywords: Rough sets, decision tables, decision reducts, geometry of reducts.

1 Introduction

In the rough set approach for decision making often are used different kinds of reducts such as reducts of information systems, decision reducts or local reducts. It is well known that the size of the set of all reducts of information (decision) systems can be huge relative to the number of attributes. Hence, the direct searching in such sets for relevant reducts can be often computationally infeasible, especially for large data tables. Hence, there is a need for developing efficient methods for extracting relevant information about reducts from data tables which could help us to induce the high quality data models such as rule based classifiers. This relevant information could help, e.g., to reduce the dimensionality of the attribute set. We present examples illustrating how such relevant information about reduct sets can be generated from information systems or decision tables. In particular, we consider a binary relation on attributes satisfied by two given attributes if and only if there does not exist reduct consisting them both. We illustrate how such a relation can be used in the reduct generation. Moreover, we prove that for any fixed natural k, there exists a polynomial in time algorithm which for a given decision table T and given k conditional attributes recognizes if there exists a decision reduct of the decision system T containing these k attributes. Using this algorithm one can, in particular, eliminate all single attributes which are not covered by any reduct of T. We also shortly discuss how the dependencies between attributes can be used in the reduct generation. Finally, we also present a list of problems related to the discussed issue.

We plan to develop a software library which could be used in preprocessing of the reduct generation.

The set of all decision reducts of a decision table T [5] contains rich information about the table T. Unfortunately, there is no polynomial algorithms for construction of the set of all reducts.

In this paper, we show that there are polynomial (in time) algorithms for obtaining of indirect but useful information about this set.

We show that for any fixed natural k, there exists a polynomial (in time) algorithm $\mathcal{A}_k$ checking, for a given decision table T and given k conditional attributes, if there exist a reduct for T covering these k attributes.

The information obtained on the basis of algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ can be represented in a simple graphical form. One can construct a graph with the set of vertices equal to the set of attributes covered by at least one reduct, and the set of edges equal to the set of all pairs of attributes which do not belong to any reduct. The degree of an attribute in this graph (the number of edges incident to this attribute) characterizes the attribute importance. The changes of this graph after adding of a new object to the decision table allow us to evaluate the degree of influence of this new object on the reduct set structure. In the paper, we consider such graphs for three real-life decision tables. Some properties of such graphs are studied in [2].

Note that there exist close analogies between results of this paper and results obtained in [1], where the following problem was considered: for a given positive Boolean function f and given subset of its variables it is required to recognize if there exists a prime implicant of dual Boolean function f^d containing these variables.

Another approach for efficient extracting from a given decision table T of indirect information about the set of all reducts and a graphical representation of the information was considered in [9]. It was shown that there exists a polynomial algorithm for constructing the so-called pairwise core graph for a given decision table T. The set of vertices of this graph is equal to the set of conditional attributes of T, and the set of edges coincides with the two element sets of attributes disjoint with the core of T (i.e., the intersection of all reducts of T) and having non-empty intersection with any reduct of T. This example is a step toward a realization of a program suggested in early 90s by Andrzej Skowron in his lectures at Warsaw University to study geometry of reducts aiming at developing tools for investigating geometrical properties of reducts in the space of all reducts of a given information system. For example, the core of a given information system can be empty but in the reduct space can exist only a few subfamilies of reducts such that the intersection of each subfamily is non-empty.

Yet another discussed in the paper method for extracting information about the reduct set from a given data table can be based on the dependencies between attributes in a given information system.

This paper is structured as follows. In Section 2, we discuss the problem of existence of reducts including a given set of attributes. The graphical representation of some information about the set of reducts is considered in Section 3. In Section 4, we discuss shortly a possible application of dependencies in the reduct generation. In Section 5, we present conclusions and a list of problems for further study. In the appendix, we present a polynomial algorithm for one of problems listed in Section 5. This algorithm was found during the final editing of the paper.

This paper is an extended version of [3].

2 On Covering of k Attribute Sets by Reducts

A *decision table* T is a finite table in which each column is labeled by a *conditional attribute*. Rows of the table T are interpreted as tuples of values of conditional attributes on some objects. Each row is labeled by a *decision* which is interpreted as the value of the *decision attribute*[1].

Let A be the set of conditional attributes (the set of names of conditional attributes) of T. We will say that a conditional attribute $a \in A$ *separates* two rows if these rows have different values at the intersection with the column labeled by a. We will say that two rows are *different* if at least one attribute $a \in A$ separates these rows. Denote by $P(T)$ the set of unordered pairs of different rows from T which are labeled by different decisions.

A subset R of the set A is called a *test* (or *superreduct*) for T if for each pair of rows from $P(T)$ there exists an attribute from R which separates rows in this pair. A test R for T is called a *reduct* for T if each proper subset of R is not a test for T. In the sequel, we deal with decision reducts but we will omit the word "decision".

Let us fix a natural number k. We consider the following *covering problem for k attributes by a reduct*: for a given decision table T with the set of conditional attributes A, a subset B of the set A, and k pairwise different attributes $a_1, \ldots, a_k \in B$ it is required to recognize if there exist a reduct R for T such that

[1] We consider uniformly both consistent and inconsistent decision tables. However, in the case of inconsistent decision table, one can use also the so called generalized decision instead of the original decision [5,6,7].

$R \subseteq B$ and $a_1, \ldots, a_k \in R$, and if the answer is "yes" it is required to construct such a reduct. We describe a polynomial in time algorithm $\mathcal{A}_k$ for the covering problem (see Algorithm 1).

For $a \in A$, we denote by $P_T(a)$ the set of pairs of rows from $P(T)$ separated by a. For $a_1, \ldots, a_k \in A$ and $a_j \in \{a_1, \ldots, a_k\}$ let

$$P_T(a_j|a_1, \ldots, a_k) = P_T(a_j) \setminus \bigcup_{i \in \{1,\ldots,k\} \setminus \{j\}} P_T(a_i).$$

For $a_1, \ldots, a_k \in A$, let

$$\mathcal{P}_T(a_1, \ldots, a_k) = P_T(a_1|a_1, \ldots, a_k) \times \ldots \times P_T(a_k|a_1, \ldots, a_k).$$

Assuming that $(\pi_1, \ldots, \pi_k) \in \mathcal{P}_T(a_1, \ldots, a_k)$, we denote by

$$D_T(B, a_1, \ldots, a_k, \pi_1, \ldots, \pi_k)$$

the set of attributes a from $B \setminus \{a_1, \ldots, a_k\}$ such that a separates rows in at least one pair of rows from the set $\{\pi_1, \ldots, \pi_k\}$. Note that

$$D_T(B, a_1, \ldots, a_k, \pi_1, \ldots, \pi_k) = \bigcup_{j=1}^{k} D_T(B, a_j, \pi_j).$$

Algorithm 1. Algorithm $\mathcal{A}_k$ for solving of the covering problem for k attributes by a reduct

```
Input  : Decision table $T$ with the set of conditional attributes $A$, $B \subseteq A$, and
         $a_1, \ldots, a_k \in B$.
Output: If there exists a reduct $R$ for $T$ such that $R \subseteq B$ and $a_1, \ldots, a_k \in R$,
         then the output is one of such reducts; otherwise, the output is "no".
construct the set $\mathcal{P}_T(a_1, \ldots, a_k)$;
for any tuple $(\pi_1, \ldots, \pi_k) \in \mathcal{P}_T(a_1, \ldots, a_k)$ do
    $R \longleftarrow B \setminus D_T(B, a_1, \ldots, a_k, \pi_1, \ldots, \pi_k)$
    if $R$ is a test for $T$ then
        while $R$ is not a reduct for $T$ do
            select $a \in R$ such that $R \setminus \{a\}$ is a test for $T$;
            $R := R \setminus \{a\}$
        end
        return $R$;
        stop
    end
end
return "no" (in particular, if $\mathcal{P}_T(a_1, \ldots, a_k) = \emptyset$, then the output is "no")
```

Using algorithm $\mathcal{A}_k$ (see Algorithm 1) first the set $\mathcal{P}_T(a_1, \ldots, a_k)$ is constructed. Next, for each tuple $(\pi_1, \ldots, \pi_k) \in \mathcal{P}_T(a_1, \ldots, a_k)$ the set

$$D_T(B, a_1, \ldots, a_k, \pi_1, \ldots, \pi_k)$$

is constructed and it is verified if the set $B \setminus D_T(B, a_1, \ldots, a_k, \pi_1, \ldots, \pi_k)$ is a test for T. It is clear that $|\mathcal{P}_T(a_1, \ldots, a_k)| \leq n^{2k}$, where n is the number of rows in T. Using this inequality and the fact that k is fixed natural number, one can prove that the algorithm $\mathcal{A}_k$ has polynomial time complexity[2]. Unfortunately, the algorithm $\mathcal{A}_k$ has a relatively high time complexity.

The considered algorithm is based on the following proposition:

Proposition 1. *Let T be a decision table with the set of conditional attributes A, $B \subseteq A$, and $a_1, \ldots, a_k \in B$. Then the following statements hold:*

1. *A reduct R for T such that $R \subseteq B$ and $a_1, \ldots, a_k \in R$ exists if and only if there exists a tuple $(\pi_1, \ldots, \pi_k) \in \mathcal{P}_T(a_1, \ldots, a_k)$ such that*
$$B \setminus D_T(B, a_1, \ldots, a_k, \pi_1, \ldots, \pi_k)$$
is a test for T.
2. *If the set $S = B \setminus D_T(B, a_1, \ldots, a_k, \pi_1, \ldots, \pi_k)$ is a test for T then each reduct Q for T, obtained from S by removing from S of some attributes, has the following properties: $a_1, \ldots, a_k \in Q$ and $Q \subseteq B$.*

Proof. Let R be a reduct for T such that $a_1, \ldots, a_k \in R$ and $R \subseteq B$. It is clear that for each $a_j \in \{a_1, \ldots, a_k\}$ there exists a pair of rows π_j from $P(T)$ such that a_j is the only attribute from the set R separating this pair. It is clear that $(\pi_1, \ldots, \pi_k) \in \mathcal{P}_T(a_1, \ldots, a_k)$ and $R \subseteq B \setminus D_T(B, a_1, \ldots, a_k, \pi_1, \ldots, \pi_k)$. Since R is a reduct for T, we conclude that $B \setminus D_T(B, a_1, \ldots, a_k, \pi_1, \ldots, \pi_k)$ is a test for T.

Let us assume that there exists a tuple $(\pi_1, \ldots, \pi_k) \in \mathcal{P}_T(a_1, \ldots, a_k)$ such that the set $S = B \setminus D_T(B, a_1, \ldots, a_k, \pi_1, \ldots, \pi_k)$ is a test for T. Let Q be a reduct for T obtained by removing some attributes from S. It is also clear that $Q \subseteq B$. Let $j \in \{1, \ldots, k\}$. Since a_j is the only attribute from the test S separating rows from π_j, we have $a_j \in Q$. Thus, $a_1, \ldots, a_k \in Q$. □

3 Graphical Representation of Information about the Set of Reducts

Let T be a decision table with the set of conditional attributes A. Let $B \subseteq A$. Using polynomial algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ one can construct a graph $G(T, B)$. The set of vertices of this graph coincides with the set of attributes $a \in B$ for each of which there exists a reduct R for T such that $R \subseteq B$ and $a \in R$. Two different vertices a_1 and a_2 of $G(T, B)$ are linked by an edge if and only if there is no a reduct R for T such that $R \subseteq B$ and $a_1, a_2 \in R$. Let us denote by $G(T)$ the graph $G(T, A)$.

Note that there exists close analogy between the graph $G(T)$ and the so-called co-occurance graph [1] for positive Boolean function f. The set of vertices of this

[2] Note that k is treated as a constant for the algorithm $\mathcal{A}_k$.

graph is equal to the set of variables of f. Two different variables are linked by an edge if and only if f has a prime implicant containing these variables.

Now, we present the results of three experiments with real-life decision tables from [4] (the first example was considered in [2]).

Example 1. [2] Let us denote by T_Z the decision table "Zoo" [4] with 16 conditional attributes $a_1, \ldots, a_{16}$ (we ignore the first attribute "animal name") and 101 rows. Only attributes $a_1, a_3, a_4, a_6, \ldots, a_{14}, a_{16}$ are vertices of the graph $G(T_Z)$. The set of reducts for T_Z is represented in Table 1. The graph $G(T_Z)$ is depicted in Fig. 1. For example, any reduct containing a_7 is disjoint with $\{a_8, a_9, a_{14}\}$.

Example 2. Let us denote by T_L the decision table "Lymphography" [4] with 18 conditional attributes $a_1, \ldots, a_{18}$ and 148 rows. Each of the considered attributes is a vertex of the graph $G(T_L)$. The graph $G(T_L)$ is depicted in Fig. 2.

Table 1. The set of reducts for the decision table T_Z ("Zoo")

$\{a_3, a_4, a_6, a_8, a_{13}\}$	$\{a_3, a_6, a_8, a_9, a_{12}, a_{13}\}$	$\{a_3, a_6, a_8, a_{13}, a_{16}\}$
$\{a_3, a_4, a_6, a_9, a_{13}\}$	$\{a_1, a_3, a_6, a_7, a_{10}, a_{12}, a_{13}\}$	$\{a_3, a_6, a_9, a_{13}, a_{16}\}$
$\{a_3, a_6, a_8, a_{10}, a_{13}\}$	$\{a_3, a_4, a_6, a_7, a_{10}, a_{12}, a_{13}\}$	$\{a_4, a_6, a_8, a_{11}, a_{13}, a_{16}\}$
$\{a_1, a_3, a_6, a_9, a_{10}, a_{13}\}$	$\{a_1, a_6, a_8, a_{10}, a_{12}, a_{13}\}$	$\{a_4, a_6, a_9, a_{11}, a_{13}, a_{16}\}$
$\{a_1, a_3, a_6, a_8, a_{11}, a_{13}\}$	$\{a_1, a_6, a_9, a_{10}, a_{12}, a_{13}\}$	$\{a_3, a_6, a_7, a_{10}, a_{11}, a_{13}, a_{16}\}$
$\{a_1, a_3, a_6, a_9, a_{11}, a_{13}\}$	$\{a_1, a_3, a_6, a_{10}, a_{13}, a_{14}\}$	$\{a_1, a_6, a_8, a_{10}, a_{11}, a_{13}, a_{16}\}$
$\{a_3, a_6, a_8, a_9, a_{11}, a_{13}\}$	$\{a_3, a_4, a_6, a_{10}, a_{13}, a_{14}\}$	$\{a_1, a_6, a_9, a_{10}, a_{11}, a_{13}, a_{16}\}$
$\{a_1, a_3, a_6, a_8, a_{12}, a_{13}\}$	$\{a_3, a_6, a_8, a_{11}, a_{13}, a_{14}\}$	$\{a_3, a_6, a_7, a_{10}, a_{12}, a_{13}, a_{16}\}$
$\{a_4, a_6, a_8, a_{12}, a_{13}\}$	$\{a_3, a_6, a_8, a_{12}, a_{13}, a_{14}\}$	$\{a_3, a_6, a_{10}, a_{13}, a_{14}, a_{16}\}$
$\{a_1, a_3, a_6, a_9, a_{12}, a_{13}\}$	$\{a_1, a_6, a_{10}, a_{12}, a_{13}, a_{14}\}$	$\{a_1, a_6, a_{10}, a_{11}, a_{13}, a_{14}, a_{16}\}$
$\{a_4, a_6, a_9, a_{12}, a_{13}\}$	$\{a_4, a_6, a_{10}, a_{12}, a_{13}, a_{14}\}$	$\{a_4, a_6, a_{10}, a_{11}, a_{13}, a_{14}, a_{16}\}$

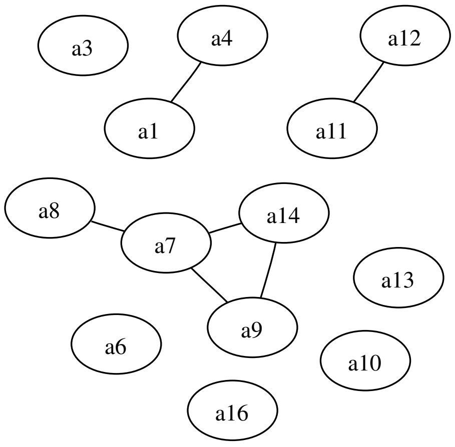


Fig. 1. Graph $G(T_Z)$ for the decision table T_Z ("Zoo")

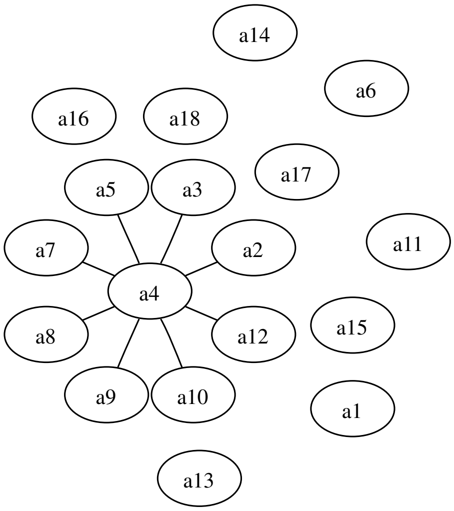


Fig. 2. Graph $G(T_L)$ for the decision table T_L ("Lymphography")

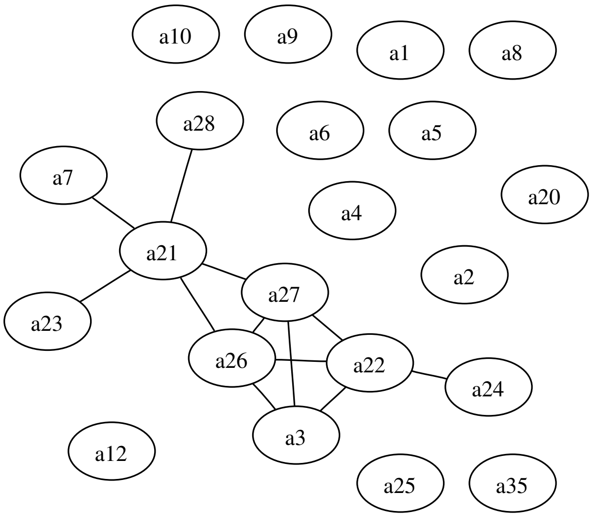


Fig. 3. Graph $G(T_S)$ for the decision table T_S ("Soybean-small")

In particular, one can observe from $G(T_L)$ that any reduct of T_L containing a_4 is disjoint with $\{a_2, a_3, a_5, a_7, a_8, a_9, a_{10}, a_{12}\}$.

Example 3. Let us denote by T_S the decision table "Soybean-small" [4] with 35 conditional attributes $a_1, \ldots, a_{35}$ and 47 rows. Only attributes $a_1, \ldots, a_{10}, a_{12}$

and $a_{20}, \ldots, a_{28}, a_{35}$ are vertices of the graph $G(T_S)$. The graph $G(T_S)$ is depicted in Fig. 3.

Some properties of graphs $G(T)$ are studied in [2].

It is shown in [2] that any undirected graph G can be represented as the graph $G(T)$ for an appropriate decision table T. However, the graph $G(T)$ can give us rich information about the set of reducts for a decision table T.

Proposition 2. *[2] Let A be a finite set of names of conditional attributes, and $G = (V, E)$ be an undirected graph, where $V \subseteq A$ is the set of vertices of G and E is the set of edges of G such that each edge of G is a two-element subset of V. Then there exists a decision table T with the set of names of conditional attributes A such that $G(T) = G$.*

Results of experiments considered in [2] show that there exists a correlation between the degrees[3] of attributes in $G(T)$ and the number of reducts of T covering these attributes (the last parameter is considered often as an attribute importance), and between changes of $G(T)$ and changes of the set of reducts for T after extending T by a new object (the changes in the set of reducts can be considered as a noticeable influence of updating of the decision table by the new object).

4 Using Dependencies in Generation of Reducts

Another important issue in data analysis is discovering dependencies between attributes in a given decision system $T = (U, C, D)$. Intuitively, a set of attributes D depends totally on a set of attributes C, denoted $C \Rightarrow D$, if the values of attributes from C uniquely determine the values of attributes from D. In other words, D depends totally on C, if there exists a functional dependency between values of C and D.

We will say that D *depends on* C to a *degree* k $(0 \leq k \leq 1)$ in T, denoted $C \Rightarrow_k D$, if

$$k = \gamma(C, D) = \frac{card(POS_C(D))}{card(U)}, \tag{1}$$

where

$$POS_C(D) = \bigcup_{X \in U/D} C_*(X),$$

called a *positive region* of the partition U/D with respect to C[4], is the set of all elements of U that can be uniquely classified to blocks of the partition U/D, by means of C.

If $k = 1$ we say that D *depends totally* on C, and if $k < 1$, we say that D *depends partially* (to *degree* k) on C. If $k = 0$ then the *positive region* of the partition U/D with respect to C is empty. The coefficient k expresses the ratio

[3] A degree of an attribute is the number of edges incident to this attribute.

[4] $C_*(X)$ denotes the C-lower approximation of X [6].

of all elements of the universe, which can be properly classified to blocks of the partition U/D, employing attributes C and will be called the *degree of the dependency.* Summing up: D is *totally* (*partially*) dependent on C, if *all* (*some*) elements of the universe U can be uniquely classified to blocks of the partition U/D, employing C. Observe, that (1) defines only one of possible measures of dependency between attributes (see, e.g., [8]).

Let us consider one very simple application of dependencies in reduct generation. One can also consider dependencies between conditional attributes in decision tables. In particular, if B, B' are subsets of conditional attributes in T then the dependency $B \Rightarrow B'$ is true in T if and only if the dependency $B \Rightarrow_1 B'$ holds in the decision system (U, B, B'). Then, in searching for decision reducts of T in which B should be included one can eliminate the attributes from B'. Obviously, if there exist two disjoint subsets B_1, B_2 of B such that $B_1 \Rightarrow B_2$ holds in T then there does not exist reduct covering B.

5 Conclusions

We have discussed some methods for generation of information from data tables which can be used in the reduct computation. In particular, we have shown that, for each natural k a polynomial algorithm $\mathcal{A}_k$ exists which for a given decision table and given k conditional attributes recognizes if there exist a decision reduct covering these k attributes. Results of computer experiments with two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ are reported. Finally, we have shortly discussed applications of dependencies between conditional attributes in the reduct generation. In our project we are building a software library which could be helpful in solving different reduct generation problems. Methods from this library could be applied as some additional tools simplifying searching for relevant reducts.

Below we present a list of exemplary problems we would like to investigate in our further study. We are interested in computational complexity of these problems and algorithms (heuristics) for solving them.

The input for each problem is a data table T representing an information or decision system. By $RED(T)$ we denote the set of all reducts of T of a given kind (e.g., decision reducts).

- *Problem 1.* Let us consider a graph $G_{RED}(T)$ with nodes equal to elements of $RED(T)$. Two reducts are linked by an edge if and only if they have non-empty intersection. We would like to estimate the number of connected components of the graph $G_{RED}(T)$ [5].
- *Problem 2.* For given thresholds $tr, k > 0$ check if there exist at least k reducts with non-empty intersection consisting at least tr attributes.

[5] During the final editing of the paper we found a polynomial algorithm for this problem solving in the case when $RED(T)$ is the set of all decision reducts (see appendix). This algorithm has a relatively high time complexity. The problem of existence of more efficient algorithms is open.

– *Problem 3.* How many maximal (with respect to the number of elements) families of reducts from $RED(T)$ exist which satisfy the condition formulated in *Problem 2*?
– *Problem 4.* Find a maximal family of pairwise disjoint reducts in $RED(T)$.
– *Problem 5.* Let us consider a discernibility function for reducts defined by $dis_T(R) = |\{R' \in RED(T) : \ R \cap R' = \emptyset\}|$ for $R \in RED(T)$. Find bounds for $f_T(n) = \max\{dis_T(R) : \ R \in RED(T) \ \& \ |R| = n\}$.
– *Problem 6.* Let us consider a binary discernibility relation on subsets of attributes defined by $B \ DIS(T) \ C$ if and only if ($B \subseteq R$ and $R \cap C = \emptyset$) or ($C \subseteq R$ and $R \cap B = \emptyset$) for some $R \in RED(T)$, where B, C are subsets of the set of (conditional) attributes of T. What are the properties of $DIS(T)$?
– *Problem 7.* Let us consider a distance between reducts defined by

$$dist_T(R, R') = |R \setminus R'| + |R' \setminus R|,$$

for $R, R' \in RED(T)$. Estimate the largest distance between reducts from $RED(T)$.
– *Problem 8.* Let us consider an incremental sequence of decision tables $T_i = (U_i, C, D)$ for $i = 1, 2, \ldots$, where $U_i \subseteq U_{i+1}$ for any i. We would like to develop methods for reasoning about changes between $RED(T_i)$ and $RED(T_{i+1})$.

In our further study we also would like to check if there exist efficient randomized algorithms for solution of the considered in the paper problem.

Acknowledgments

The research has been partially supported by the grant N N516 368334 from Ministry of Science and Higher Education of the Republic of Poland and by the grant "Decision support – new generation systems" of Innovative Economy Operational Programme 2008-2012 (Priority Axis 1. Research and development of new technologies) managed by Ministry of Regional Development of the Republic of Poland.

The authors are grateful to Marcin Piliszczuk for performing of experiments and for his suggestion allowing us to improve algorithms $\mathcal{A}_k$.

References

1. Boros, E., Gurvich, V., Hammer, P.L.: Dual subimplicants of positive Boolean functions. Optimization Methods and Software 10, 147–156 (1998)
2. Moshkov, M., Piliszczuk, M.: Graphical representation of information on the set of reducts. In: Yao, J., Lingras, P., Wu, W.-Z., Szczuka, M.S., Cercone, N.J., Ślęzak, D. (eds.) RSKT 2007. LNCS (LNAI), vol. 4481, pp. 372–378. Springer, Heidelberg (2007)
3. Moshkov, M., Skowron, A., Suraj, Z.: On covering attribute sets by reducts. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 175–180. Springer, Heidelberg (2007)

4. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI Repository of machine learning databases. University of California, Irvine, Department of Information and Computer Sciences (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
5. Pawlak, Z.: Rough Sets – Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht (1991)
6. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Information Sciences 177(1), 3–27 (2007)
7. Pawlak, Z., Skowron, A.: Rough sets and Boolean reasoning. Information Sciences 177(1), 41–73 (2007)
8. Ślęzak, D.: Approximate entropy reducts. Fundamenta Informaticae 53, 365–387 (2002)
9. Wróblewski, J.: Pairwise cores in information systems. In: Ślęzak, D., Wang, G., Szczuka, M.S., Düntsch, I., Yao, Y. (eds.) RSFDGrC 2005. LNCS (LNAI), vol. 3641, pp. 166–175. Springer, Heidelberg (2005)

Appendix: Comparison of Graphs $G_{RED}(T)$ and $G'(T)$

In this section, we consider a polynomial algorithm for solving of *Problem 1* in the case of decision reducts.

Let T be a decision table. We denote by $RED(T)$ the set of all decision reducts for T. The set of nodes of the graph $G_{RED}(T)$ coincides with $RED(T)$. Two reducts are linked by an edge if and only if they have non-empty intersection. Graph $G'(T)$ is the complement of the graph $G(T)$. The set of nodes of the graph $G'(T)$ coincides with the set $A_{RED}(T) = \bigcup_{R \in RED(T)} R$ of all conditional attributes of T each of which belongs to at least one decision reduct for T. Two attributes are linked by an edge if and only if there exists a decision reduct for T containing both these attributes.

We show that graphs $G_{RED}(T)$ and $G'(T)$ have the same number of connected components.

Proposition 3. *Let T be a decision table. Then the number of connected components of the graph $G_{RED}(T)$ is equal to the number of connected components of the graph $G'(T)$.*

Proof. Let $B_1, \ldots, B_n$ be all connected components of the graph $G_{RED}(T)$. For $i = 1, \ldots, n$, we denote by A_i the set of attributes contained in reducts belonging to B_i. It is clear that $A_1 \cup \ldots \cup A_n = A_{RED}(T)$ and $A_i \cap A_j = \emptyset$ for any $i, j \in \{1, \ldots, n\}$, $i \neq j$. For $i = 1, \ldots, n$, we denote by $G'(T, A_i)$ the subgraph of the graph $G'(T)$ generated by nodes from A_i.

Let us prove that $G'(T, A_1), \ldots, G'(T, A_n)$ are all connected components of the graph $G'(T)$. To this end we must show that, for any $i, j \in \{1, \ldots, n\}$, $i \neq j$, the following statements hold:

1. Any two nodes from A_i are connected by a path in the graph $G'(T)$.
2. Any node from A_i and any node from A_j are not linked by an edge.

Let $|A_i| = 1$. Then the first statement holds. The unique attribute from A_i forms a reduct. This attribute can not belong to any other reduct. Therefore, the second statement holds too.

Let $|A_i| > 1$. Let us consider arbitrary different nodes a and a' from A_i. Then there are reducts R and R' from B_i such that $a \in R$ and $a' \in R'$. If $R = R'$ then a and a' are linked by an edge. Let us assume that $R \neq R'$. We consider a shortest path $\alpha = R, R_1, \ldots, R_m, R'$ in $G_{RED}(T)$ connecting the reducts R and R'. We set $R_0 = R$ and $R_{m+1} = R'$. For $t = 0, \ldots, m$, we choose an attribute $a_t \in R_t \cap R_{t+1}$. It is clear that $R_1, \ldots, R_m$ belong to B_i. Therefore, $a_t \in A_i$ for $t = 0, \ldots, m$. Let us consider the sequence

$$\beta = a, a_0, \ldots, a_m, a'.$$

Notice that it is possible that $a = a_0$ or $a_m = a'$. Since α is a shortest path connecting R and R', we have $a_0 \neq a_1 \neq a_2 \neq \ldots \neq a_m$. It is clear that $a, a_0 \in R_0 = R$, $a_0, a_1 \in R_1$, ..., $a_{m-1}, a_m \in R_m$, $a_m, a' \in R_{m+1} = R'$. Therefore, the sequence β forms a path connecting a an a' in $G'(T)$. Thus, the first statement holds.

Let us assume that there exist a node $a_1 \in A_i$ and a node $a_2 \in A_j$ which are linked by an edge. Then there exists a reduct R such that $a_1, a_2 \in R$, which is impossible since $R \in B_i$ and $R \in B_j$. Thus, the second statement holds. Hence, we obtain that $G'(T, A_1), \ldots, G'(T, A_n)$ are all connected components of the graph $G'(T)$. □

Proposition 3 allows us to solve *Problem 1* (for the case of decision reducts for a decision table T) in the following way: using polynomial algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ we construct the graph $G'(T)$ and find (in polynomial time) the number of connected components in this graph. The obtained number is equal to the number of connected components in the graph $G_{RED}(T)$.

Unfortunately, algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ have a relatively high time complexity. The problem of existence of more efficient algorithms for *Problem 1* solving is open.

Example 4. Let us consider decision tables T_Z ("Zoo"), T_L ("Lymphography") and T_S ("Soybean-small") discussed in Examples 1-3. Simple analysis of graphs $G(T_Z)$, $G(T_L)$ and $G(T_S)$, which are the complements of the graphs $G'(T_Z)$, $G'(T_L)$ and $G'(T_S)$, shows that each of graphs $G'(T_Z)$, $G'(T_L)$ and $G'(T_S)$ has exactly one connected component. Therefore, each of graphs $G_{RED}(T_Z)$, $G_{RED}(T_L)$ and $G_{RED}(T_S)$ has exactly one connected component.

Context Algebras, Context Frames, and Their Discrete Duality*

Ewa Orłowska[1] and Ingrid Rewitzky[2]

[1] National Institute of Telecommunications, Warsaw, Poland
E.Orlowska@itl.waw.pl

[2] Department of Mathematical Sciences, University of Stellenbosch, South Africa
rewitzky@sun.ac.za

Abstract. The data structures dealt with in formal concept analysis are referred to as contexts. In this paper we study contexts within the framework of discrete duality. We show that contexts can be adequately represented by a class of sufficiency algebras called context algebras. On the logical side we define a class of context frames which are the semantic structures for context logic, a lattice-based logic associated with the class of context algebras. We prove a discrete duality between context algebras and context frames, and we develop a Hilbert style axiomatization of context logic and prove its completeness with respect to context frames. Then we prove a duality via truth theorem showing that both context algebras and context frames provide the adequate semantic structures for context logic. We discuss applications of context algebras and context logic to the specification and verification of various problems concerning contexts such as implications (attribute dependencies) in contexts, and derivation of implications from finite sets of implications.

Keywords: Duality, duality via truth, representation theorem, formal concept analysis, context, concept, attribute dependency, implication.

1 Introduction

A fundamental structure arising in formal concept analysis (FCA) [7,21] is that of a 'context'. In this paper we will consider this notion within the framework of what we refer to as discrete duality. While a classical duality, such as that of, for example, Stone [19] and Priestley [18], includes a representation of a class of algebras in terms of a topological structure, a discrete duality includes a representation for a class of algebras in terms of the relational structures that provide the frame semantics (or equivalently, Kripke-style semantics) of the lattice-based logic associated with the class of algebras. The frame semantics is given in terms of a relational structure without a (non-discrete) topology which

* Ewa Orłowska acknowledges partial support from the Polish Ministry of Science and Higher Education grant N N206 399134, and Ingrid Rewitzky acknowledges partial support from National Research Foundation of South Africa. Both authors acknowledge support from the Poland/SA Bilateral Collaboration Programme.

explains the name of this type of duality. General principles of establishing a discrete duality are presented in [14]. In this paper we elaborate in detail a discrete duality for the structures arising in connection with problems considered in formal concept analysis.

First, we show that contexts can be adequately represented by an axiomatic and signature extension of the class of sufficiency algebras [6], referred to as context algebras. The lattice-based logic associated with the class of context algebras we call context logic. Context algebras are Boolean algebras with a pair of sufficiency operators forming a Galois connection. The sufficiency operators of context algebras are the abstract counterparts of the maps of extent and intent determined by a context. On the logical side we define a class of context frames which are the semantic structures for context logic. We prove a discrete duality between context algebras and context frames. This duality is established in two steps: we show a representation theorem for context algebras such that a representation algebra is constructed from a context frame and is shown to contain (up to isomorphism) a given algebra as a subalgebra; next we show a representation theorem for context frames such that a representation frame is constructed from a context algebra and is shown to contain (up to isomorphism) a given context frame as a substructure. A discrete duality for sufficiency algebras is presented in [6], see also [12]. It extends Jónsson-Tarski duality [10] for Boolean algebras with normal and additive operators to Boolean algebras with a sufficiency operator. The discrete duality proved in the present paper extends this result further to context algebras. Discrete dualities for distributive lattices with operators forming a Galois connection and also some other similar connections are developed in [15]. Next, we develop a Hilbert-style axiomatization of context logic and prove its completeness with respect to context frames.

Second, we discuss applications of context algebras and context logic to the specification and verification of various problems concerning contexts and concepts. We consider three groups of problems: first, problems related to the verification of whether a pair 'set of objects, set of attributes' is a formal concept having some properties; second, attribute dependencies in information systems which are closely related to contexts; third, implications in contexts and derivation of implications from finite sets of implications. We indicate that the tasks from all these three groups can be specified within the framework of context algebras and context logic presented in this paper. The deduction tools of context logic and the theories of context algebras can then be used for solving these problems.

2 Context Algebras and Frames

Central in formal concept analysis is the notion of a *Galois connection* between two types of entities. Algebraically this may be captured by two maps e and i over a Boolean algebra, and relationally by a relation between the two types of entities. In this section we formalise this in the notions of context algebra and context frame.

Definition 1. *A context algebra* $(W, \vee, \wedge, \neg, 0, 1, e, i)$ *is a Boolean algebra* $(W, \vee, \wedge, \neg, 0, 1)$ *endowed with unary operators* e, i *satisfying, for any* $a, b \in W$,

(AC1) $g(a \vee b) = g(a) \wedge g(b)$ *for* $g = e, i$
(AC2) $g(0) = 1$ *for* $g = e, i$
(AC3) $a \leq e(i(a))$
(AC4) $a \leq i(e(a))$.

An operator $g : W \to W$ satisfying (AC1) and (AC2) is called a *sufficiency operator*, see [6]. It follows that the sufficiency operators e and i are antitone and form a Galois connection, that is,

$$a \leq i(b) \quad \text{iff} \quad b \leq e(a), \qquad \text{for any } a, b \in W.$$

From this Galois connection we can separate the two types of entities and the relation between them thereby deriving a formal context arising in formal concept analysis. From a context algebra $\mathcal{A} = (W, \vee, \wedge, \neg, 0, 1, e, i)$ we may define the formal context $\mathcal{C}_{\mathcal{A}} = (G_{\mathcal{A}}, M_{\mathcal{A}}, I_{\mathcal{A}})$, where $G_{\mathcal{A}} = \{o \mid \exists a, o \leq e(a)\}$, $M_{\mathcal{A}} = \{a \mid \exists o, a \leq i(o)\}$ and $I_{\mathcal{A}} = \{(o, a) \mid o \leq e(a)\} = \{(o, a) \mid a \leq i(o)\}$. Then $G_{\mathcal{A}} = \mathrm{dom}(I_{\mathcal{A}})$ and $M_{\mathcal{A}} = \mathrm{ran}(I_{\mathcal{A}})$.

Lemma 1. *Let* $\mathcal{A} = (W, \vee, \wedge, \neg, 0, 1, e, i)$ *be a context algebra. The sufficiency operators* e *and* i *of* $\mathcal{A}$ *are the mappings of extent and intent determined by a formal context* $\mathcal{C}_{\mathcal{A}} = (G_{\mathcal{A}}, M_{\mathcal{A}}, I_{\mathcal{A}})$, *in the sense that, for any* $a, o \in W$,

$$o \leq e(a) \text{ iff } o \in e_{\mathcal{A}}(\{a\}) = \{o \mid oI_{\mathcal{A}}a\}$$

and

$$a \leq i(o) \text{ iff } a \in i_{\mathcal{A}}(\{o\}) = \{a \mid oI_{\mathcal{A}}a\}.$$

On the other hand, given a formal context $\mathcal{C} = (G, M, I)$, we may define a context algebra $(W_{\mathcal{C}}, e_{\mathcal{C}}, i_{\mathcal{C}})$ where $W_{\mathcal{C}} = 2^{G \cup M}$, $e_{\mathcal{C}} = [\![I]\!]$ and $i_{\mathcal{C}} = [\![I^{-1}]\!]$, where for any $A \in W_{\mathcal{C}}$ and $T = I, I^{-1}$,

$$[\![T]\!](A) = \{x \in G \cup M \mid \forall y,\ y \in A \ \Rightarrow\ xTy\}.$$

Lemma 2. *Let* $\mathcal{C} = (G, M, I)$ *be a formal context. The mappings of extent and intent determined by* $\mathcal{C}$ *are the sufficiency operators of the context algebra* $\mathcal{A}_{\mathcal{C}} = (W_{\mathcal{C}}, e_{\mathcal{C}}, i_{\mathcal{C}})$, *that is,* $e = e_{\mathcal{C}}$ *and* $i = i_{\mathcal{C}}$.

Theorem 1

(a) *If a formal context* $\mathcal{C} = (G, M, I)$ *satisfies* $G = \mathrm{dom}(I)$ *and* $M = \mathrm{ran}(I)$, *then* $\mathcal{C} = \mathcal{C}_{\mathcal{A}_{\mathcal{C}}}$.
(b) *If a context algebra* $\mathcal{A} = (W, \vee, \wedge, \neg, 0, 1, e, i)$ *is complete and atomic and such that* $W = 2^X$ *where* $X = \{o \mid o \in e(\{a\})\} \cup \{a \mid a \in i(\{o\})\}$, *then* $\mathcal{A} = \mathcal{A}_{\mathcal{C}_{\mathcal{A}}}$.

Definition 2. *A context frame* $\mathcal{F} = (X, R, S)$ *is a non-empty set* X *endowed with binary relations* R *and* S *such that* $S = R^{-1}$.

Although relation S is definable from R, the setting with two relations enables us to avoid any relational-algebraic structure (in this case the operation of converse of a relation) in the language of context logic developed in Section 3. In this way the intended object language is singular, the required constraint is formulated only in the definition of its semantics, that is, in the metalanguage.

From a context frame $\mathcal{F} = (X, R, S)$ we may define the formal context $\mathcal{C}_\mathcal{F} = (G_\mathcal{F}, M_\mathcal{F}, I_\mathcal{F})$, where $G_\mathcal{F} = dom(R)$, $M_\mathcal{F} = ran(R)$, and $I_\mathcal{F} = R$.

Lemma 3. *Let $\mathcal{F} = (X, R, S)$ be a context frame. The sufficiency operators determined by $\mathcal{F}$ are the mappings of extent and intent determined by a formal context $\mathcal{C}_\mathcal{F} = (G_\mathcal{F}, M_\mathcal{F}, I_\mathcal{F})$, that is, $[\![R]\!] = e_\mathcal{F}$ and $[\![S]\!] = i_\mathcal{F}$.*

On the other hand, given a formal context $\mathcal{C} = (G, M, I)$, we may define a context frame $\mathcal{F}_\mathcal{C} = (X_\mathcal{C}, R_\mathcal{C}, S_\mathcal{C})$, where $X_\mathcal{C} = G \cup M$, $R_\mathcal{C} = I$ and $S_\mathcal{C} = I^{-1}$.

Lemma 4. *Let $\mathcal{C} = (G, M, I)$ be a formal context. The mappings of extent and intent determined by $\mathcal{C}$ are the sufficiency operators determined by the context frame $\mathcal{F}_\mathcal{C} = (X_\mathcal{C}, R_\mathcal{C}, S_\mathcal{C})$, that is, $e = [\![R_\mathcal{C}]\!]$ and $i = [\![S_\mathcal{C}]\!]$.*

Theorem 2

(a) *If a formal context $\mathcal{C} = (G, M, I)$ satisfies $G = \mathrm{dom}(I)$ and $M = \mathrm{ran}(I)$, then $\mathcal{C} = \mathcal{C}_{\mathcal{F}_\mathcal{C}}$.*
(b) *If a context frame $\mathcal{F} = (X, R, S)$ satisfies $X = \mathrm{dom}(R) \cup \mathrm{ran}(R)$, then $\mathcal{F} = \mathcal{F}_{\mathcal{C}_\mathcal{F}}$.*

We now establish a discrete duality between context algebras and context frames. First, we show that from any context frame we can define a context algebra. Let (X, R, S) be a context frame. Then the binary relations R and S over X induce sufficiency operators over 2^X, namely, $e^c : 2^X \to 2^X$ defined, for any $A \in 2^X$, by

$$e^c(A) = [\![R]\!](A) = \{x \in X \mid \forall y \in X,\ y \in A \ \Rightarrow\ xRy\},$$

and $i^c : 2^X \to 2^X$ defined for any $A \in 2^X$, by

$$i^c(A) = [\![S]\!](A) = \{x \in X \mid \forall y \in X,\ y \in A \ \Rightarrow\ xSy\}.$$

Thus,

Definition 3. *Let (X, R, S) be a context frame. Then its complex algebra is the powerset Boolean algebra with sufficiency operators $(2^X, \cup, \cap, -, \emptyset, X, e^c, i^c)$.*

Theorem 3. *The complex algebra of a context frame is a context algebra.*

Proof: The operators e^c and i^c are sufficiency operators as shown in [12]. By way of example we show that (AC3) is satisfied. Let $A \subseteq X$, we show that $A \subseteq e^c(i^c(A))$. Let $x \in X$ and suppose that $x \in A$ but $x \notin e^c(i^c(A))$. It follows that there is $y_0 \in i^c(A)$ such that $(x, y_0) \notin R$. By definition of i^c, for every $z \in A$, $y_0 S z$. In particular, taking z to be x, we have $y_0 S x$. Since $S = R^{-1}$, we have $x R y_0$, a contradiction. The proof of (AC4) is similar. □

Next we show that any context algebra in turn gives rise to a context frame. In the case of a sufficiency operator g over a powerset Boolean algebra 2^X, a relation r_g over X may be defined, as in [6], by

$$xr_gy \quad \text{iff} \quad x \in g(\{y\}), \qquad \text{for any } x, y \in X.$$

In general, as in [12], we invoke Stone's representation theorem and then define, from each sufficiency operator $g : W \to W$, a binary relation R_g over the family $\mathcal{X}(W)$ of prime filters of W by

$$FR_gG \quad \text{iff} \quad g(G) \cap F \neq \emptyset, \quad \text{for any } F, G \in \mathcal{X}(W)$$

where for $A \subseteq W$, $g(A) = \{g(a) \in W \mid a \in A\}$. It is an easy exercise to show that R_g is an extension of r_g. Thus,

Definition 4. *The* canonical frame *of a context algebra* $(W, \vee, \wedge, \neg, 0, 1, e, i)$ *is the relational structure* $(\mathcal{X}(W), R^c, S^c)$*, where* $\mathcal{X}(W)$ *is the family of prime filters of* W*,* $R^c = R_e$ *and* $S^c = R_i$*.*

Theorem 4. *The canonical frame of a context algebra is a context frame.*

Proof: We show that $(R^c)^{-1} \subseteq S^c$. Let $(F, G) \in (R^c)^{-1}$. Then $(G, F) \in R^c$, that is $e(F) \cap G \neq \emptyset$. Hence, there is some a_0 such that $a_0 \in G$ and $a_0 \in e(F)$. Take b_0 to be $i(a_0)$. Then $b_0 \in i(G)$ and $b_0 \in i(e(F))$. Now $i(e(F)) \subseteq F$ since if $a \in i(e(F))$ then $a = i(e(x))$ for some $x \in F$, so, by (AC4) and since F is up-closed, $a = i(e(x)) \in F$. Thus $b_0 \in i(G) \cap F$, that is, $i(G) \cap F \neq \emptyset$. The proof of the other inclusion is similar. □

Let $(W, \vee, \wedge, \neg, 0, 1, e, i)$ be a context algebra. Then

$$(2^{\mathcal{X}(W)}, \cup, \cap, -, \emptyset, \mathcal{X}(W), e^c, i^c)$$

is the complex algebra of the canonical frame $(\mathcal{X}(W), R^c, S^c)$ of the original context algebra. The relationship between these algebras is captured by the Stone mapping $h : W \to 2^{\mathcal{X}(W)}$ defined, for any $a \in W$, by

$$h(a) = \{F \in \mathcal{X}(W) \mid a \in F\}.$$

This mapping is an embedding and preserves operators i and e over W. That is,

Theorem 5. *For any context algebra* $(W, \vee, \wedge, \neg, 0, 1, e, i)$ *and any* $a \in W$*,*

$$h(e(a)) = e^c(h(a)) \qquad \text{and} \qquad h(i(a)) = i^c(h(a)).$$

Proof: We give the proof for e; that for i is similar. We need to show, for any $a \in W$ and any $F \in \mathcal{X}(W)$, that

$$e(a) \in F \quad \text{iff} \quad \forall G \in \mathcal{X}(W),\ a \in G \ \Rightarrow\ e(G) \cap F \neq \emptyset.$$

Assume $a \in G$ and $e(G) \cap F = \emptyset$. Then $e(a) \in e(G)$ and hence $e(a) \notin F$. On the other hand, assume $e(a) \notin F$. Let a dual of e, denoted e^d, be defined, for any $b \in W$, by $e^d(b) = -e(-b)$. Consider the set $Z_e = \{b \in W \mid e^d(b) \notin F\}$. Let F' be the filter generated by $Z_e \cup \{a\}$, that is, $F' = \{b \in W \mid \exists a_1, \ldots, a_n \in Z_e, \quad a_1 \wedge \ldots \wedge a_n \wedge a \leq b\}$. Then F' is proper. Suppose otherwise. Then for some $a_1, \ldots, a_n \in Z_e, \quad a_1 \wedge \ldots \wedge a_n \wedge a = 0$, that is, $a \leq -(a_1 \wedge \ldots \wedge a_n) = -a_1 \vee \ldots \vee -a_n$. Since e is antitone, $e(-a_1 \vee \ldots \vee -a_n) \leq e(a)$. Thus $e(-a_1) \wedge \ldots \wedge e(-a_n) \leq e(a)$, that is, $-e^d(a_1) \wedge \ldots \wedge -e^d(a_n) \leq e(a)$. By definition of Z_e we have $e^d(a_1), \ldots, e^d(a_n) \notin F$ so $-e^d(a_1), \ldots, -e^d(a_n) \in F$. Since F is a filter, $-e^d(a_1) \wedge \ldots \wedge -e^d(a_n) \in F$ and hence $e(a) \in F$ which contradicts the original assumption. So, by ([4], p188), there is a prime filter G containing F'. Since $a \in F'$, $a \in G$ and hence $G \in h(a)$. Also $e(G) \cap F = \emptyset$ since if there is some $b \in W$ with $b \in e(G)$ and $b \in F$, then $b = e(c)$ for some $c \in G$ and thus $e(c) \in F$, so $e^d(-c) \notin F$ hence $-c \in Z_e \subseteq F' \subseteq G$ and thus $c \notin G$, which is a contradiction. □

On the other hand, let (X, R, S) be a context frame. The $(\mathcal{X}(2^X), R^c, S^c)$ is the canonical frame of the complex algebra $(2^X, \cup, \cap, -, \emptyset, X, e^c, i^c)$ of the original context frame. The relationship between these frames is captured by the mapping $k : X \to \mathcal{X}(2^X)$ defined, for any $x \in X$, by $k(x) = \{A \in 2^X \mid x \in A\}$. It is easy to show that k is well-defined and an embedding. All that remains is to show that k preserves structure, that is,

Theorem 6. *For any context frame (X, R, S) and any $x, y \in X$,*

$$xRy \text{ iff } k(x)R^c k(y) \qquad \text{and} \qquad xSy \text{ iff } k(x)S^c k(y).$$

Proof: We give the proof for R; that for S is similar. Note, for any $x, y \in X$,

$$\begin{aligned} k(x)R^c k(y) \quad &\text{iff} \quad [\![R]\!](k(y)) \cap k(x) \neq \emptyset \\ &\text{iff} \quad \exists A \in 2^X,\ y \in A \ \wedge\ \forall z \in X, z \in A \Rightarrow xRz. \end{aligned}$$

Suppose $k(x)R^c k(y)$ does not hold. Let $A = \{y\}$. Then $y \in A$ and hence, for some $z \in A$, xRz does not hold. Therefore, $z = y$ and xRy does not hold. Suppose $k(x)R^c k(y)$. Let $A = \{y\}$. Then $y \in A$ and hence xRy. □

Therefore, we have a discrete duality between context algebras and context frames.

Theorem 7

(a) *Every context algebra can be embedded into the complex algebra of its canonical frame.*
(b) *Every context frame can be embedded into the canonical frame of its complex algebra.*

3 Context Logic

In order to extend the duality established in Theorem 7 to a Duality via Truth as considered in [13], we need a logical language presented in [8].

Definition 5. *Let* LC *be a modal language extending the language of classical propositional calculus, that is, its formulas are built from propositional variables taken from an infinite denumerable set V and the constants true* (1) *and false* (0)*, with the classical propositional operations of negation* ($\neg$)*, disjunction* ($\vee$)*, conjunction* ($\wedge$)*, and with two unary operators* $[\![]\!]_1$ *and* $[\![]\!]_2$*. As usual,* $\rightarrow$ *and* $\leftrightarrow$ *are definable:*

$$\phi \rightarrow \psi := \neg\phi \vee \psi \quad \text{and} \quad \phi \leftrightarrow \psi := (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi).$$

Also the constants are definable: $0 := \phi \wedge \neg\phi$ and $1 := \phi \vee \neg\phi$*. We define* $\langle\!\langle\rangle\!\rangle_i$*, by* $\langle\!\langle\rangle\!\rangle_i\phi := \neg[\![]\!]_i\neg\phi$ *(for* $i = 1, 2$*). Let* For *denote the set of all formulae of* LC*.*

Within LC the conditions on a context algebra can be captured, using a Hilbert-style axiomatisation, as follows:

Axioms:

(LC0) Axioms of the classical propositional calculus (see eg [16])
(LC1) $[\![]\!]_i(\phi \vee \psi) \leftrightarrow [\![]\!]_i\phi \wedge [\![]\!]_i\psi$ (for $i = 1, 2$)
(LC2) $[\![]\!]_i 0 = 1$ (for $i = 1, 2$)
(LC3) $\phi \rightarrow [\![]\!]_1[\![]\!]_2\phi$
(LC4) $\phi \rightarrow [\![]\!]_2[\![]\!]_1\phi$.

Rules of inference: modus ponens and

$$\frac{\phi \rightarrow \psi}{[\![]\!]_i\psi \rightarrow [\![]\!]_i\phi} \qquad \text{(for } i = 1, 2\text{)}.$$

If ϕ is obtained from the axioms by repeated applications of the inference rules, then ϕ is called a *theorem* of LC, written $\vdash \phi$. Axioms (LC3) and (LC4) reflect the fact that the two sufficiency operators form a Galois connection. Some logics arising from a Galois connection are also considered in [20].

The semantics of context logic LC is based on context frames (X, R, S) where X is a non-empty set endowed with binary relations R and S such that $S = R^{-1}$. A LC-model based on a context frame (X, R, S) is a system $M = (X, R, S, m)$, where $m : V \cup \{0, 1\} \rightarrow 2^X$ is a meaning function such that $m(p) \subseteq X$ for $p \in V$, $m(0) = \emptyset$, $m(1) = X$. The satisfaction relation $\models$ is defined as follows, where $M, x \models \phi$ means that the state x satisfies formula ϕ in model M:

$$\begin{array}{rll}
M, x \models p & \text{iff} & x \in m(p), \text{ for every } p \in V \\
M, x \models \phi \vee \psi & \text{iff} & M, x \models \phi \text{ or } M, x \models \psi \\
M, x \models \phi \wedge \psi & \text{iff} & M, x \models \phi \text{ and } M, x \models \psi \\
M, x \models \neg\phi & \text{iff} & \text{not } M, x \models \phi \\
M, x \models [\![]\!]_1\phi & \text{iff} & \forall y \in X,\ M, y \models \phi \text{ implies } xRy \\
M, x \models [\![]\!]_2\phi & \text{iff} & \forall y \in X,\ M, y \models \phi \text{ implies } xSy.
\end{array}$$

From now on we shall write $[\![R]\!]$ and $[\![S]\!]$ instead of $[\![]\!]_1$ and $[\![]\!]_2$, respectively. A notion of truth of formulae based on the LC semantics is defined as usual. A formula $\phi \in \mathsf{LC}$ is true in a context model M, written $M \models \phi$, whenever for every $x \in X$ we have $M, x \models \phi$. A formula $\phi \in \mathsf{LC}$ is true in a context frame (X, R, S) iff ϕ is true in every model based on this frame. And finally a formula $\phi \in \mathsf{LC}$ is valid in the logic LC, called LC-valid and written $\models \phi$, iff it is true in every context frame.

Theorem 8. (Soundness) *For any* LC*-formula* $\phi \in \mathsf{For}$*, if* ϕ *is a theorem of* LC *then* ϕ *is* LC*-valid.*

Proof: Proving soundness is an easy task — it involves showing that the axioms of LC are LC-valid and the rules preserve LC-validity. □

Following a technique due to Rasiowa [16] to prove completeness, we need some constructions and lemmas. The relation $\approx$ defined on the set For of formulae of LC by:

$$\phi \approx \psi \quad \text{iff} \quad \vdash (\phi \leftrightarrow \psi)$$

is an equivalence relation compatible with the operations $\vee, \wedge, \neg, [\![R]\!], [\![S]\!], 0, 1$. This induces a quotient algebra

$$\mathcal{A}_{\approx} = (\mathsf{For}|_{\approx}, \cup, \cap, -, 0_{\approx}, 1_{\approx}, [\![R]\!]_{\approx}, [\![S]\!]_{\approx})$$

where $\mathsf{For}|_{\approx}$ is the family of equivalence classes of $\approx$ and, for any $\phi, \psi \in \mathsf{For}$,

$$|\phi| \cup |\psi| = |\phi \vee \psi| \quad |\phi| \cap |\psi| = |\phi \wedge \psi| \quad -|\phi| = |\neg\phi| \quad 0_{\approx} = |\phi \wedge \neg\phi| \quad 1_{\approx} = |\phi \vee \neg\phi|$$

$$[\![T]\!]_{\approx}|\phi| = |[\![T]\!]\phi| \quad (\text{for } T = R, S).$$

Then for the definable connectives, $\rightarrow$ and $\leftrightarrow$, we postulate:

$$|\phi| \rightarrow |\psi| = |\phi \rightarrow \psi| \quad \text{and} \quad |\phi| \leftrightarrow |\psi| = |\phi \leftrightarrow \psi|.$$

Lemma 5

(a) $\mathcal{A}_{\approx}$ *is a non-degenerate* (*i.e. at least two-element*) *context algebra.*
(b) *For any* $\phi, \psi \in \mathsf{For}$, $|\phi| \leq_{\approx} |\psi|$ iff $\vdash \phi \rightarrow \psi$.
(c) *For any* $\phi \in \mathsf{For}$, $\vdash \phi$ iff $|\phi| = 1_{\approx}$.
(d) *For any* $\phi \in \mathsf{For}$, $|\neg\phi| \neq \emptyset$ iff not $\vdash \phi$.

Proof: For (a), we show that the Lindenbaum algebra satisfies conditions (AC1)-(AC4). For (AC1) and (AC2) we consider $[\![R]\!]_{\approx}$; the proofs for $[\![S]\!]_{\approx}$ are similar.

For (AC1), $[\![R]\!]_{\approx}(|\phi| \cup |\psi|) = [\![R]\!]_{\approx}(|\phi \vee \psi|) = |[\![R]\!](\phi \vee \psi)| = |[\![R]\!]\phi \wedge [\![R]\!]\psi|$ $= |[\![R]\!]\phi| \cap |[\![R]\!]\psi| = [\![R]\!]_{\approx}|\phi| \cap [\![R]\!]_{\approx}|\psi|$.

For (AC2), $[\![R]\!]_{\approx}0_{\approx} = [\![R]\!]_{\approx}|\phi \wedge \neg\phi| = |[\![R]\!](\phi \wedge \neg\phi)| = |[\![R]\!](0)| = |1| =$ $|\phi \vee \neg\phi| = 1_{\approx}$.

For (AC3), we have to show that for any formula ϕ, $|\phi| \subseteq [\![R]\!]_{\approx}[\![S]\!]_{\approx}|\phi|$. By (LC3) and the definition of the operators in the Lindenbaum algebra $\mathcal{A}_{\approx}$ we have $|\phi| \subseteq |[\![R]\!][\![S]\!]\phi| = [\![R]\!]_{\approx}[\![S]\!]_{\approx}|\phi|$. The proof of (AC4) is similar.

For (b), $|\phi| \leq_{\approx} |\psi|$ iff $|\phi| \cup |\psi| = |\psi|$ iff $|\phi \vee \psi| = |\psi|$ iff $\vdash \phi \vee \psi \leftrightarrow \psi$ iff $\vdash \phi \rightarrow \psi$.

For (c), assume $\vdash \phi$. Since $\vdash \phi \to ((\phi \to \phi) \to \phi)$, by applying modus ponens, we get $\vdash (\phi \to \phi) \to \phi$. By (b), $|\phi \to \phi| \leq_{\approx} |\phi|$, hence $1_{\approx} = |\phi|$. Now assume that $|\phi| = 1_{\approx}$. Since $1_{\approx} = |\phi \to \phi|$, we have $|\phi \to \phi| \leq_{\approx} |\phi|$. By (b), $\vdash (\phi \to \phi) \to \phi$. Since $\vdash \phi \to \phi$, by applying modus ponens we get $\vdash \phi$.

For (d), we use (b) and the fact that $|\phi| = 1_{\approx}$ iff $|\neg\phi| = 0_{\approx} = \emptyset$. □

Consider the canonical frame $(\mathcal{X}(\mathcal{A}_{\approx}), R_{\approx}, S_{\approx})$ of the algebra $\mathcal{A}_{\approx}$ where, for any $F, G \in \mathcal{X}(\mathcal{A}_{\approx})$,

$$FT_{\approx}G \quad \text{iff} \quad [\![T]\!]_{\approx}(G) \cap F \neq \emptyset, \quad \text{for } T = R, S.$$

Definition 6. *The canonical* LC*-model based on the canonical context frame* $(\mathcal{X}(\mathcal{A}_{\approx}), R_{\approx}, S_{\approx})$ *of* $\mathcal{A}_{\approx}$ *is a system* $M_{\approx} = (\mathcal{X}(\mathcal{A}_{\approx}), R_{\approx}, S_{\approx}, m_{\approx})$*, where the meaning function* $m_{\approx} : V \cup \{0,1\} \to 2^{X(\mathcal{A}_{\approx})}$ *is defined, for any* $p \in V \cup \{0,1\}$ *and any* $F \in \mathcal{X}(\mathcal{A}_{\approx})$*, by*

$$F \in m_{\approx}(p) \qquad \text{iff} \qquad |p| \in F.$$

The mapping $m_{\approx}$ *extends homomorphically to all* LC*-formulae, that is, for any* $\phi, \psi \in$ For *and* $T = R, S$*,*

$$m_{\approx}(\neg\phi) = -m_{\approx}(\phi),\ m_{\approx}(\phi \vee \psi) = m_{\approx}(\phi) \cup m_{\approx}(\psi),\ m_{\approx}([\![T]\!]\phi) = [\![T]\!]_{\approx}(m_{\approx}(\phi)).$$

Consider the mapping $h_{\approx} : \mathcal{A}_{\approx} \to 2^{\mathcal{X}(\mathcal{A}_{\approx})}$ defined as on page 216, that is, for any $|\phi| \in \mathcal{A}_{\approx}$,

$$h_{\approx}(|\phi|) = \{F \in \mathcal{X}(\mathcal{A}_{\approx}) \mid |\phi| \in F\}.$$

We know from Theorem 5 that $h_{\approx}$ preserves the operations (all operations, not only the sufficiency operator).

Lemma 6. *Let* $\mathcal{M}_{\approx}$ *be the canonical* LC*-model based on the canonical context frame* $(\mathcal{X}(\mathcal{A}_{\approx}), R_{\approx}, S_{\approx})$*. Then, for any* $F \in \mathcal{X}(\mathcal{A}_{\approx})$ *and any* LC *formula* ϕ*,*

$$M_{\approx}, F \models \phi \text{ iff } F \in h_{\approx}(|\phi|).$$

Proof: For this we use structural induction on LC-formulae ϕ. By definition, for any basic formula $p \in V$,

$$\mathcal{M}_{\approx}, F \models p \quad \text{iff} \quad F \in m_{\approx}(p) \quad \text{iff} \quad |p| \in F \text{ iff } F \in h_{\approx}(|p|).$$

Assume as induction hypothesis that the claim holds for $\phi, \psi \in$ For. We consider the cases where θ is $\phi \vee \psi$ and $[\![R]\!]\phi$; the other cases are similar.

$$\begin{array}{lll}
\mathcal{M}_{\approx}, F \models \phi \vee \psi & \text{iff} & \mathcal{M}_{\approx}, F \models \phi \text{ or } \mathcal{M}_{\approx}, F \models \psi \\
& \text{iff} & F \in h_{\approx}(|\phi|) \text{ or } F \in h_{\approx}(|\psi|) \\
& \text{iff} & |\phi| \in F \text{ or } |\psi| \in F \\
& \text{iff} & |\phi| \cup |\psi| \in F \quad F \text{ is a prime filter} \\
& \text{iff} & |\phi \vee \psi| \in F \\
& \text{iff} & F \in h_{\approx}(|\phi \vee \psi|).
\end{array}$$

$$\begin{array}{rll} \mathcal{M}_\approx, F \models [\![R]\!]\phi & \text{iff} & \forall G \in \mathcal{X}(\mathcal{A}_\approx),\ \mathcal{M}_\approx, G \models \phi \text{ implies } FR_\approx G \\ & \text{iff} & \forall G \in \mathcal{X}(\mathcal{A}_\approx),\ G \in h_\approx(|\phi|) \text{ implies } FR_\approx G \\ & \text{iff} & F \in [\![R_\approx]\!](h_\approx(|\phi|)) \\ & \text{iff} & F \in h_\approx([\![R]\!]_\approx|\phi|) \quad \text{by Theorem 5} \\ & \text{iff} & F \in h_\approx(|[\![R]\!]\phi|). \end{array}$$

This completes the proof. □

Thus, since $h_\approx$ is an embedding that preserves operations $[\![T]\!]$ (for $T = R, S$), we have the following truth lemma.

Lemma 7. *For any $F \in \mathcal{X}(\mathcal{A}_\approx)$ and any* LC *formula ϕ,*

$$M_\approx, F \models \phi \qquad \text{iff} \qquad |\phi| \in F.$$

Theorem 9. (Completeness) *For any* LC*-formula $\phi \in$* For*, if ϕ is* LC*-valid then ϕ is a theorem of* LC*.*

Proof: Take any LC formula ϕ such that $\models \phi$. Suppose that $\vdash \phi$ does not hold. Then, by Lemma 5(d), $|\neg\phi| \neq \emptyset$. So there exists $F \in \mathcal{X}(\mathcal{A}_\approx)$ such that $|\neg\phi| \in F$. Thus, by Lemma 7, for some $F \in \mathcal{X}(\mathcal{A}_\approx)$, $M_\approx, F \models \neg\phi$, Hence, by definition of $\models$, ϕ is not true in $M_\approx$, which contradicts the assumption that ϕ is LC-valid. □

With this logical language we can extend Theorem 7 to a Duality via Truth, in the sense of [13]. Let $\mathsf{Alg}_{\mathsf{LC}}$ denote the class of context algebras, and $\mathsf{Frm}_{\mathsf{LC}}$ denote the class of context frames. As described above the class $\mathsf{Frm}_{\mathsf{LC}}$ of context frames provides a frame semantics for LC. The class $\mathsf{Alg}_{\mathsf{LC}}$ of context algebras provides an algebraic semantics for LC. Let $(W, \vee, \wedge, \neg, 0, 1, e, i)$ be a context algebra. A valuation on W is a function $v : V \to W$ which assigns elements of W to propositional variables and extends homomorphically to all the formulas of LC, that is

$$v(\neg\phi) = \neg v(\phi),\ v(\phi \vee \psi) = v(\phi) \vee v(\psi),\ \ v([\![R]\!]\phi) = e(v(\phi)),\ \ v([\![S]\!]\phi) = i(v(\phi)).$$

The notion of truth determined by this semantics is as follows. A formula ϕ in LC is true in an algebra $(W, \vee, \wedge, \neg, 0, 1, e, i)$ whenever $v(\phi) = 1$ for every v in W. A formula $\phi \in$ LC is true in the class $\mathsf{Alg}_{\mathsf{LC}}$ of context algebras iff it is true in every context algebra in $\mathsf{Alg}_{\mathsf{LC}}$.

Theorem 10. *A formula $\phi \in$* LC *is true in every model based on a context frame (X, R, S) iff ϕ is true in the complex algebra $(2^X, e^c, i^c)$ defined from that frame.*

Proof: Let (X, R, S) be any context frame. The result is established by taking the meaning function m on any model (X, R, S, m) based on (X, R, S) to coincide with the valuation function on the complex algebra $(2^X, e^c, i^c)$ of (X, R, S). □

Finally, we prove the Duality via Truth theorem between context algebras and context frames.

Theorem 11. *A formula $\phi \in \mathsf{LC}$ is true in every context algebra in $\mathsf{Alg}_{\mathsf{LC}}$ iff ϕ is true in every context frame in $\mathsf{Frm}_{\mathsf{LC}}$.*

Proof: Assume that ϕ is true in all context algebras. In particular, ϕ is true in all the complex algebras of the context frames. Since every context frame has its corresponding complex algebra, by Theorem 10 the required condition follows. For reverse implication, we prove the contrapositive. Assume that for some context algebra W and a valuation v in W $v(\phi) \neq 1$. Consider the canonical frame $\mathcal{X}(W)$ of W. By the representation theorem there is an embedding $h : W \to 2^{\mathcal{X}(W)}$. It follows that $h(v(\phi)) \neq 1$. Consider a model M based on $\mathcal{X}(W)$ such that $m(p) = h(v(p))$. By induction on the complexity of formulas we can show that for every formula ϕ, $m(\phi) = m(h(\phi))$. Hence, ϕ is not true in M. □

4 Applications for Formal Concept Analysis

In this section we show that context algebras and context logic can be used for the specification and verification of various problems concerning contexts and concepts from formal concept analysis.

4.1 Intents, Extents and Operations of Concepts

Let (G, M, I), where $I \subseteq G \times M$, be a context. The Galois connection underlying the notion of context algebra and context frame allows the identification of certain pairs (O, A) where $O \in 2^G$, $A \in 2^M$. Namely, those that are closed in the sense that $i(O) = A$ and $e(A) = O$. That is, given a set O of objects the map i of the Galois connection identifies all the features which they have in common, and given a set A of features the map e of the Galois connection identifies all the objects which they have in common. Such object-feature pairs are called *formal concepts*. The sets $i(O)$ and $e(A)$ are called the *intent* and *extent* of the concept (O, A), respectively.

As explained in [22] care needs to be taken when defining operations of join, meet and complement for concepts in order to ensure the result is again a formal concept. Accordingly, given two formal concepts (O, A) and (O', A'), their join $\vee$ and meet $\wedge$ are defined respectively by:

$$(O, A) \vee (O', A') = (e(i(O \cup O')), A \cap A')$$
$$(O, A) \wedge (O', A') = (O \cap O', i(e(A \cup A'))).$$

The corresponding order $\leq$ on the set of formal concepts is defined, for formal concepts (O, A) and (O', A'), by

$$(O, A) \leq (O', A') \quad \text{iff} \quad O \subseteq O' \quad (\text{or equivalently, iff } A' \subseteq A).$$

With respect to this order, the smallest formal concept is $(e(M), M)$ and the largest is $(G, i(G))$.

For the complement of a formal concept (O, A), two complements are considered. The one is generated by the set complement $\overline{O}$ of the extent O and the other generated by the set complement $\overline{A}$ of the intent A. Namely, for a formal concept (O, A), its weak negation is defined

$$\neg(O, A) = (e(i(\overline{O})), i(\overline{O}))$$

and its weak opposition is defined by

$$\sim (O, A) = (e(\overline{A}), i(e(\overline{A}))).$$

Note that $\neg$ captures contradictory opposite, in the sense that, for example, positive and negative, and cold and hot are contradictory opposites. On the other hand $\sim$ captures contrary opposite, in the sense that, for example, moist and dry, and cold and warm are contrary opposites.

We now characterize the notions of intent and extent, and the operations of join, meet, weak negation, weak opposition for concepts in terms of LC-formulae. For this, take any $O_i \in 2^G$ (for $i = 1, 2, 3$) and any $A_i \in 2^M$ (for $i = 1, 2, 3$). Suppose that p_i is the propositional variable representing O_i (for $i = 1, 2, 3$), and that q_i is the propositional variable representing A_i (for $i = 1, 2, 3$).

Problems concerning extents:

– $O_1 \subseteq G$ is an extent of some concept
  iff $O_1 = e(i(O_1))$
  iff $p_1 \leftrightarrow [\![R]\!][\![S]\!]p_1$ is true in the models such that the meaning of p_1 is O_1.
– (O_1, A_1) is the unique concept of which O_1 is an extent
  iff $A_1 = i(O_1)$ and $O_1 = e(i(O_1))$
  iff $q_1 \leftrightarrow [\![S]\!]p_1 \wedge p_1 \leftrightarrow [\![R]\!][\![S]\!]p_1$ is true in the models such that the meanings of p_1 and q_1 are O_1 and A_1, respectively.

Problems concerning intents:

– $A_1 \subseteq M$ is an intent of some concept
  iff $A_1 = i(e(A_1))$
  iff $q_1 \leftrightarrow [\![S]\!][\![R]\!]q_1$ is true in the models such that the meaning of q_1 is A_1.
– (O_1, A_1) is the unique concept of which A_1 is an intent
  iff $O_1 = e(A_1)$ and $A_1 = i(e(A_1))$
  iff $p_1 \leftrightarrow [\![R]\!]q_1 \wedge q_1 \leftrightarrow [\![S]\!][\![R]\!]q_1$ is true in the models such that the meanings of p_1 and q_1 are O_1 and A_1, respectively.

Problems concerning operations on concepts:

– A formal concept (O_1, A_1) is the join of two concepts (O_2, A_2) and (O_3, A_3)
  iff $O_1 = e(i(O_2 \cup O_3))$ and $A_1 = A_2 \cap A_3$
  iff $(p_1 \leftrightarrow [\![R]\!][\![S]\!](p_2 \vee p_3)) \;\wedge\; (q_1 \leftrightarrow q_2 \wedge q_3)$ is true in the models such that, for $i = 1, 2, 3$, the meanings of p_i and q_i are O_i and A_i, respectively.
– A formal concept (O_1, A_1) is the meet of two concepts (O_2, A_2) and (O_3, A_3)
  iff $O_1 = O_2 \cap O_3$ and $A_1 = i(e(A_2 \cup A_3))$
  iff $(p_1 \leftrightarrow p_2 \wedge p_3) \;\wedge\; (q_1 \leftrightarrow [\![S]\!][\![R]\!](q_2 \vee q_3))$ is true in the models such that, for $i = 1, 2, 3$, the meanings of p_i and q_i are O_i and A_i, respectively.

– A formal concept (O_1, A_1) is a weak negation of some concept (O_2, A_2) iff $O_1 = e(i(\overline{O_2}))$ and $A_1 = i(\overline{O_2})$ iff $(p_1 \leftrightarrow [\![R]\!][\![S]\!]\neg p_2) \ \wedge \ (q_1 \leftrightarrow [\![S]\!]\neg p_2)$ is true in the models such that, for $i = 1, 2$, the meanings of p_i and q_i are O_i and A_i, respectively.
– A formal concept (O_1, A_1) is a weak opposition of some concept (O_2, A_2) iff $O_1 = e(\overline{A_2})$ and $A_1 = i(e(\overline{A_2}))$ iff $(p_1 \leftrightarrow [\![R]\!]\neg q_2) \ \wedge \ (q_1 \leftrightarrow [\![S]\!][\![R]\!]\neg q_2)$ is true in the models such that, for $i = 1, 2$, the meanings of p_i and q_i are O_i and A_i, respectively.

It follows that the reasoning tools of the context logic LC can be used for verification of the properties of concepts listed above, among others. A dual tableau deduction system for the logic LC is presented in [8].

4.2 Dependencies of Attributes

Discovering dependencies in sets of data is an important issue addressed in various theories, in particular in rough set theory [17] and in formal concept analysis [7]. Typically, in an information system objects are described in terms of some attributes and their values. The queries to an information system often have the form of a request for finding a set of objects whose sets of attribute values satisfy some conditions. This leads to the notion of information relation determined by a set of attributes. Let $a(x)$ and $a(y)$ be sets of values of an attribute a of the objects x and y, respectively. We may want to know a set of those objects from an information system whose sets of values of all (or some) of the attributes from a subset A of attributes are equal (or disjoint, or overlap etc.). To represent such queries we define, first, information relations on the set of objects. Some examples, defined in [5], include similarity relation, indiscernibility relations, forward inclusion, backward inclusion, negative similarity, incomplementarity relation. In the rough set-based approach the most fundamental information relation is indiscernibility and its weaker version, namely, similarity.

Let M be a set of attributes and G a set of objects. Given an attribute $a \in M$, the similarity relation $\mathsf{sim}(a) \subseteq G \times G$ is defined, for any $x, y \in G$, by

$$(x, y) \in \mathsf{sim}(a) \quad \text{iff} \quad a(x) \cap a(y) \neq \emptyset$$

and the indiscernibility relation $\mathsf{ind}(a) \subseteq G \times G$ is defined, for any $x, y \in G$, by

$$(x, y) \in \mathsf{ind}(a) \quad \text{iff} \quad a(x) = a(y).$$

These relations can be extended to any subset A of attributes by quantifying over A:

$$\begin{array}{lll} (x, y) \in \mathsf{sim}(A) & \text{iff} \quad a(x) \cap a(y) \neq \emptyset & \text{for all (some) } a \in A. \\ (x, y) \in \mathsf{ind}(A) & \text{iff} \quad a(x) = a(y) & \text{for all (some) } a \in A. \end{array}$$

Relations defined with the universal (existential) quantifier are referred to as strong (weak) relations.

Attribute dependencies, introduced in [2], express a constraint between two sets of attributes. Such constraints have been used to exclude from an information system data inappropriate for a particular application. An example of an attribute dependency involving single sets A and B of attributes is a *functional dependency* $A \to B$. Typically, a functional dependency is based on an information relation, for example,

$$\begin{array}{lll} A \to_{\mathsf{sim}} B & \text{iff} & \mathsf{sim}(A) \subseteq \mathsf{sim}(B) \\ A \to_{\mathsf{ind}} B & \text{iff} & \mathsf{ind}(A) \subseteq \mathsf{ind}(B). \end{array}$$

Some attribute dependencies involve combinations of attributes. For example, a *multi-valued dependency* $A \to\to B$ between sets A and B of attributes is defined by

$$A \to\to B \qquad \text{iff} \qquad \mathsf{ind}(A) \subseteq \mathsf{ind}(A \cup B); \mathsf{ind}(M/(A \cup B))),$$

where for relations R and S over a universe U their composition $R;S$ is defined, for any $x, y \in U$, by $xR;Sy$ iff for some $z \in U$ xRz and zSy.

A representation of attribute dependencies in terms of relations generated by equivalence relations $\mathsf{ind}(A)$ is presented in [3].

In the next two theorems we will characterise these notions within the framework of context algebras. For this we need the following observations. For each $A \subseteq M$, $\mathsf{ind}(A) \subseteq G \times G$ and ind may be viewed as a binary relation of type $M \times (G \times G)$. Then $[\![\mathsf{ind}(A)]\!] : 2^G \to 2^G$ is given, for any $Q \subseteq G$, by

$$[\![\mathsf{ind}(A)]\!](Q) = \{x \in G \mid \forall y \in G,\ y \in Q \ \Rightarrow\ (x,y) \in \mathsf{ind}(A)\},$$

and $[\![\mathsf{ind}]\!] : 2^{G \times G} \to 2^M$ is given, for any $R \subseteq G \times G$, by

$$[\![\mathsf{ind}]\!](R) = \{a \in M \mid \forall (x,y) \in G \times G,\ (x,y) \in R \ \Rightarrow\ (x,y) \in \mathsf{ind}(a)\}.$$

Also, $\mathsf{ind}^{-1} \subseteq (G \times G) \times M$, so $[\![\mathsf{ind}^{-1}]\!] : 2^M \to 2^{G \times G}$ is given, for any $A \subseteq M$, by

$$[\![\mathsf{ind}^{-1}]\!](A) = \{(x,y) \mid \forall a \in M,\ a \in A \ \Rightarrow\ (x,y) \in \mathsf{ind}(a)\}.$$

Similarly, for sim.

Theorem 12. *For any $A, B \in 2^M$,*

$$\begin{array}{lll} A \to_{\mathsf{ind}} B & \text{iff} & B \subseteq [\![\mathsf{ind}]\!][\![\mathsf{ind}^{-1}]\!](A) \\ A \to_{\mathsf{sim}} B & \text{iff} & B \subseteq [\![\mathsf{sim}]\!][\![\mathsf{sim}^{-1}]\!](A) \end{array}$$

Proof: For any $A \in 2^M$ and any $b \in M$,

$$\begin{array}{lll} A \to_{\mathsf{ind}} b & \text{iff} & \mathsf{ind}(A) \subseteq \mathsf{ind}(b) \\ & \text{iff} & \forall x, y,\ A \subseteq \mathsf{ind}^{-1}((x,y)) \ \Rightarrow\ (x,y) \in \mathsf{ind}(b) \\ & \text{iff} & \forall x, y,\ (x,y) \in [\![\mathsf{ind}^{-1}]\!](A) \ \Rightarrow\ (x,y) \in \mathsf{ind}(b) \\ & \text{iff} & [\![\mathsf{ind}^{-1}]\!](A) \subseteq \mathsf{ind}(b) \\ & \text{iff} & b \in [\![\mathsf{ind}]\!][\![\mathsf{ind}^{-1}]\!](A) \end{array}$$

Now $A \rightarrow_{\mathsf{ind}} B$ holds if, for all $b \in B$, $A \rightarrow_{\mathsf{ind}} b$ holds, hence the result follows. Similarly, for sim. □

For any binary relation $R \subseteq X \times Y$, operators $[R] : 2^Y \rightarrow 2^X$ and $\langle R \rangle : 2^Y \rightarrow 2^X$ may be defined in terms of the sufficiency operator as follows:

$$[R]Q = [\![-R]\!](-Q) \qquad \langle R \rangle Q = -[\![-R]\!]Q, \quad \text{for any } Q \in 2^Y.$$

Theorem 13. *For any $A, B \in 2^M$,*

$$A \rightarrow\!\!\rightarrow B \quad \text{iff} \quad \forall y \in G,\ y \in [(\mathsf{ind}(A))^{-1}]\,\langle \mathsf{ind}(A \cup B)\rangle\,[\![\mathsf{ind}]\!](M/\ (A \cup B))(\{y\}).$$

Proof: For this it suffices to show that for any $A, B, C \in 2^M$,

$$\mathsf{ind}(A) \subseteq \mathsf{ind}(B);\mathsf{ind}(C) \quad \text{iff} \quad \forall y \in G,\ y \in [(\mathsf{ind}(A))^{-1}]\,\langle \mathsf{ind}(B)\rangle\,[\![\mathsf{ind}(C)]\!](\{y\}).$$

$$\begin{array}{ll}
 & \mathsf{ind}(A) \subseteq \mathsf{ind}(B);\mathsf{ind}(C) \\
\text{iff} & \forall x, \forall y,\ (x,y) \in \mathsf{ind}(A) \ \Rightarrow\ \exists z, (x,z) \in \mathsf{ind}(B) \ \wedge\ (z,y) \in \mathsf{ind}(C) \\
\text{iff} & \forall x, \forall y,\ (x,y) \in \mathsf{ind}(A) \ \Rightarrow\ \exists z, (x,z) \in \mathsf{ind}(B) \ \wedge\ z \in [\![\mathsf{ind}(C)]\!](\{y\}) \\
\text{iff} & \forall y, \forall x,\ (x,y) \in \mathsf{ind}(A) \ \Rightarrow\ x \in \langle \mathsf{ind}(B)\rangle[\![\mathsf{ind}(C)]\!](\{y\}) \\
\text{iff} & \forall y,\ y \in [(\mathsf{ind}(A))^{-1}]\langle \mathsf{ind}(B)\rangle[\![\mathsf{ind}(C)]\!](\{y\}).
\end{array}$$
□

Relationships between Galois connections and dependencies of attributes are also studied in [9]. It is shown there that every Galois connection between two complete lattices determines an Armstrong system of functional dependencies.

4.3 Implications

In the representation of data of an information system as a formal context, (many-valued) attributes are refined into several (single-valued) features which are essentially attribute-value pairs. For example, the attribute colour may be refined to the attribute-value pair (colour, green) which corresponds to the feature being of colour green. Each object determines an object-concept (O, A) where A is the set of features of the given object, and O is the set of all objects having features in A. On the other hand, each feature determines a feature-concept (O, A) where O is the set of objects having the given feature, and A is the set of all features of objects in O.

Constraints between two sets of features are usually called *implications.* An *implication* $A \rightarrow B$ between sets A and B of features holds in a context (G, M, I) iff $e(A) \subseteq e(B)$, meaning that each object in G having all the features from A has all the features from B. An implication $A \rightarrow B$ between sets A and B of attributes is *trivial*, if $B \subseteq A$. As a consequence of the connections established in Section 2 between context algebras and formal contexts, we have

Theorem 14. *Let (G, M, I) be a context. For any $A, B \in 2^M$,*

$$A \rightarrow B \ \text{ iff } \ [\![I]\!](A) \subseteq [\![I]\!](B).$$

Hence an implication $A \rightarrow B$ holds in a context (G, M, I) iff

$$\forall g \in G,\ (\forall a \in A,\ gIa) \Rightarrow (\forall b \in B,\ gIb).$$

This is the definition of an association rule [1] used in data mining and therefore within our framework we have established a connection between implications in formal concept analysis and association rules. Moreover, the above provides an alternative to the relational characterisation, considered in [11], in terms of a so-called association relation.

Taking into account support and confidence, an association rule is defined, in [1], to be a constraint, denoted by $r : A \rightarrow B$, between sets A and B of attributes where $A, B \neq \emptyset$, $A \cap B = \emptyset$, and the support and the confidence of $r : A \rightarrow B$ are defined respectively to be

$$\mathsf{supp}(r) = \frac{|e(A \cup B)|}{|G|} \qquad \text{and} \qquad \mathsf{conf}(r) = \frac{|e(A \cup B)|}{|e(A)|},$$

where $|X|$ denotes the cardinality of a set X.

The set of association rules holding in a formal context (G, M, R) given $\mathsf{minsupp}$ and $\mathsf{minconf}$ is

$$AR = \{r : A \rightarrow B/A \mid A \subset B \subseteq M\ \wedge\ \mathsf{supp}(r) \geq \mathsf{minsupp}\ \wedge\ \mathsf{conf}(r) \geq \mathsf{minconf}\}.$$

If $\mathsf{conf}(r) = 1$ then r is called an *exact*. If $\mathsf{supp}(r) = \mathsf{supp}(A \cup B) = \mathsf{supp}(A)$ and $\mathsf{conf}(r) = 1$ then r is called a *deterministic* association rule. Otherwise it is an *approximate association rule.*

Since the notions of support and confidence are defined in terms of the extent operator e which can be characterised in terms of a sufficiency operator, we have the following characterisation within our framework of this notion of deterministic rule.

Theorem 15. *Let (G, M, I) be a context. If $r : A \rightarrow B/A$ is a deterministic association rule then $[\![I]\!](A \cup B) = G = [\![I]\!](A) = [\![I]\!](B)$.*

Proof: Assume $A, B \in 2^M$ are non-empty. If $r : A \rightarrow B/A$ is a deterministic association rule then $A \subset B$ and $|[\![I]\!](B)| = |[\![I]\!](A)|$ and $|[\![I]\!](A \cup B)| = |G|$. Hence $A \subset B$ and $|[\![I]\!](A)| = |[\![I]\!](B)| = |[\![I]\!](A \cup B)| = |G|$. Thus $[\![I]\!](A) = [\![I]\!](B) = [\![I]\!](A \cup B) = G$. $\square$

Therefore, this notion of deterministic association rule is a special type of implication arising in formal concept analysis.

5 Conclusion

The aim of this paper has been to present a framework, based on discrete duality, for representing contexts from formal concept analysis. For contexts we established a discrete duality between context algebras and context frames, the latter being the frame semantics for context logic. In addition, we motivate the usefulness of the associated context logic for reasoning about properties of formal concepts, of attribute dependencies, and of implications.

This paper builds on earlier work in a number of ways. First, the discrete duality between context algebras and context frames extends an observation that intent and extent operators of a context are sufficiency operators on Boolean algebras, and provides another application of the duality via truth framework of [13]. Second, the uniform characterizations of often independently studied attribute dependencies and implications are new and allow for their comparison with the notion of association rule [1] used in data mining. As a consequence, the associated context logical techniques may be used for verifying typical problems, such as satisfaction and logical implication, of attribute dependencies and/or association rules.

A number of challenges remain. For example: to extend the connections established in Subsection 4.3 and develop an approach based on the presented framework for mining association rules. Perhaps further questions will occur to the reader.

Acknowledgement. The authors would like to thank the anonymous referees for helpful comments and suggestions.

References

1. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Buneman, P., Jajodia, S. (eds.) Proceedings of the 1993 ACM SIGMOD International Conference on Managements of Data, pp. 207–216 (1993)
2. Armstrong, W.W.: Dependency structures of database relationships. Information Processing Letters 7, 580–583 (1974)
3. Buszkowski, W., Orłowska, E.: Indiscernibility based formalization of dependencies in information systems. In: Orłowska, E. (ed.) Incomplete Information: Rough Set Analysis, pp. 293–315. Physica-Verlag, Heidelberg (1997)
4. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order. Cambridge University Press, Cambridge (1990)
5. Demri, S.P., Orłowska, E.S.: Incomplete Information: Structure, Inference, Complexity. EATCS Monographs in Theoretical Computer Science. Springer, Heidelberg (2002)
6. Düntsch, I., Orłowska, E.: Beyond modalities: Sufficiency and mixed algebras. In: Orłowska, E., Szałas, A. (eds.) Relational Methods in Algebra, Logic, and Computer Science, pp. 277–299. Physica-Verlag, Heidelberg (2001)
7. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg (1999)
8. Golińska-Pilarek, J., Orłowska, E.: Relational Reasoning in Formal Concept Analysis. In: Proceedings of the 2007 IEEE International Conference on Fuzzy Systems, London, pp. 1048–1053. IEEE Press, Los Alamitos (2007)
9. Järvinen, J.: Pawlak's Information Systems in Terms of Galois Connections and Functional Dependencies. Fundamenta Informaticae 75, 315–330 (2007)

10. Jónson, B., Tarski, A.: Boolean algebras with operators, Part II. American Journal of Mathematics 74, 127–162 (1952)
11. MacCaul, W.: A tableaux for the implication problem for association rules. In: Orłowska, E., Szalas, A. (eds.) Relational Methods in Computer Science Applications, pp. 73–91. Springer-Physica Verlag, Heidelberg (2001)
12. Orłowska, E., Rewitzky, I., Düntsch, I.: Relational semantics through duality. In: Proceedings of the 8th International Conference on Relational Methods in Computer Science and 3rd International Conference on Applications of Kleene Algebras. LNCS, vol. 2939, pp. 17–32. Springer, Heidelberg (2006)
13. Orłowska, E., Rewitzky, I.: Duality via Truth: Semantic frameworks for lattice-based logics. Logic Journal of the IGPL 13(4), 467–490 (2005)
14. Orłowska, E., Rewitzky, I.: Discrete duality and its application to reasoning with incomplete information. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 51–56. Springer, Heidelberg (2007)
15. Orłowska, E., Rewitzky, I.: Algebras for Galois-style connections and their discrete duality. Fuzzy Sets and Systems (submitted, 2008)
16. Rasiowa, H., Sikorski, R.: The Mathematics of Metamathematics. Polish Scientific Publishers, Warsaw (1963)
17. Pawlak, Z.: Rough Sets - Theoretical Aspects of Reasoning about Data. Kluwer, Dordrecht (1991)
18. Priestley, H.A.: Representation of distributive lattices by means of ordered Stone spaces. Bulletin of the London Mathematical Society 2, 186–190 (1970)
19. Stone, M.H.: The theory of representations for Boolean algebras. Transactions of the American Mathematical Society 40, 37–111 (1936)
20. Von Karger, B.: Temporal algebra. Mathematical Structures in Computer Science 8, 277–320 (1995)
21. Wille, R.: Restructuring lattice theory: An approach based on hierarchies of concepts. In: Rival, I. (ed.) Ordered sets, NATO Advanced Studies Institute, vol. 83, pp. 445–470. Reidel, Dordrecht (1982)
22. Wille, R.: Boolean concept logic. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS (LNAI), vol. 1867, pp. 317–331. Springer, Heidelberg (2000)

A Study in Granular Computing: On Classifiers Induced from Granular Reflections of Data[⋆]

Lech Polkowski[1] and Piotr Artiemjew[2]

[1] Polish-Japanese Institute of Information Technology
Koszykowa 86, 02008 Warszawa, Poland
polkow@pjwstk.edu.pl

[2] University of Warmia and Mazury, Zolnierska 14, 10560 Olsztyn, Poland
artem@matman.uwm.edu.pl

To the memory of Professor Zdzisław Pawlak

Abstract. Granular Computing as a paradigm in the area of Approximate Reasoning/Soft Computing, goes back to the idea of L. A. Zadeh (1979) of computing with collections of similar entities. Both fuzzy and rough set theories are immanently occupied with granules as atomic units of knowledge are inverse images of fuzzy membership functions in the first and indiscernibility classes in the other set theory.

Research on granulation in the framework of rough set theory has started soon after Zadeh's program manifest (T.Y. Lin, L.Polkowski, Qing Liu, A.Skowron, J.Stepaniuk, Y.Y.Yao) with various tools from general theory of binary relations (T.Y.Lin, Y.Y.Yao), rough mereology (L.Polkowski, A.Skowron), approximation spaces (A. Skowron and J. Stepaniuk), logics for approximate reasoning (L.Polkowski, M. Semeniuk-Polkowska, Qing Liu).

The program of granular computing requires that granules formed from entities described by data should enter computing process as elementary units of computation; this program has been pursued in some aspects of reasoning under uncertainty like fusion of knowledge, rough–neural computing, many agent systems.

In this work, granules of knowledge are exploited in tasks of classification of data. This research is a follow–up on the program initiated by the first author in plenary talks at IEEE International Conferences on Granular Computing in Beijing, 2005, and Atlanta, 2006. The idea of this program consists in granulating data and creating a granular data set (called the granular reflection of the original data set); due to expected in the process of granulation smoothing of data, eliminating of outliers, and averaging of attribute values, classification on the basis of granular data is expected to be of satisfactory quality, i.e., granulation should preserve information encoded in data to a satisfactory degree. It should be stressed, however, that the proposed process of building a granular structure involves a few random procedures (factoring attributes through a

[⋆] This work is an extended and augmented with new results version of the plenary talk by the first author at RSEISP07 International Conference [32].

granule, selection of a granular covering of the universe of objects) which makes it difficult for a rigorous analysis.

It is the aim of this work to verify the program of granular classification on the basis of experiments with real data.

Granules of knowledge are in this work defined and computed on lines proposed by Polkowski in teh framework of rough mereology: it does involve usage of similarity measures called rough inclusions along with techniques of mereological theory of concepts. In consequence, definitions of granules are invariant with respect to the choice of the underlying similarity measure.

Granules of knowledge enter the realm of classification problems in this work from a three–fold perspective: first, granulated data sets give rise to new data sets on which classifiers are tested and the results are compared to results obtained with the same classifiers on the original data sets; next, granules of training objects as well as granules of rules obtained from the training set vote for value of decision at a test object; this is repeated with granules of granular reflections of granules and with granules of rules obtained from granulated data sets. Finally, the voting is augmented with weights resulting from the distribution of attribute values between the test object and training objects.

In the first case, the rough inclusion based on Hamming's metric is applied (or, equivalently, it is the rough inclusion produced from the archimedean t–norm of Łukasiewicz); in the last two cases, rough inclusions are produced on the basis of residual implications induced from continuous t–norms of Łukasiewicz, the product t–norm, and the minimum t–norm, respectively.

In all cases results of experiments on chosen real data sets, most often used as a test data for rough set methods, are very satisfactory, and, in some cases, offer results better than many other rough set based classification methods.

Keywords: rough inclusion, mereology, granulation of knowledge, granular decision systems, data classification.

1 Introduction: Rough Computing, Rough Inclusions

Rough set theory, proposed by Zdzisław Pawlak [20],[19], deals with analysis of uncertainty in terms of indiscernibility of objects (the *Leibnizian Identity of Indiscernibles Principle*, see [50]), resulting from description of objects in terms of a finite number of attributes which usually are not adequate to the task of discriminating among objects, otherwise discernible by, e.g., numbering, time of arrival, etc.etc..

1.1 Indiscernibility, Granulation by Indiscernibles

Concepts formed within a given knowledge are defined in terms of *indiscernibility classes*; knowledge is represented as a pair (U, R) called an *approximation space*, where U is a set of *objects*, and R is a collection of *elementary classifications*, being in the simplest case *equivalence relations* on the set U.

Objects are represented in this setting by means of their *information sets* of the form $\{(r, [u]_r) : r \in R\}$ for each $u \in U$, where $[u]_r$ denotes the equivalence class of u relative to r.

A *concept* is any subset of the set U. By a R_0*–proper entity*, we mean any entity e constructed from objects in U and relations in $R_0 \subseteq R$ such that its action $e \cdot u$ on each object $u \in U$ satisfies the condition: if $(u, v) \in r$ for each $r \in R_0$ then $e \cdot u = e \cdot v$; proper concepts are also called *exact*, concepts not proper are called *rough*.

Most often, and practically always in real applications, approximation spaces are expressed in the form of *information systems*, i.e., pairs of the form $I = (U, A)$, where U is a set of objects and A is a finite set of *attributes*, where each $a \in A$ is a mapping $a : U \to V_a$ from U into the *a–value set* V_a. The collection R of equivalence relations is now induced as the collection $IND(A) = \{Ind(a) : a \in A\}$ of *indiscernibility relations*, each relation $Ind(a) = \{(u, v) : u, v \in U, a(u) = a(v)\}$.

For any subset $B \subseteq A$, the relation $Ind(B) = \bigcap_{a \in B} Ind(a)$ is the relation of *B–indiscernibility*. A concept $X \subseteq U$ is *B–exact* if and only if it is a proper concept with respect to the pair $(U, \{Ind(a) : a \in B\})$.

A particular case of an information system is a *decision system*, i.e., a pair $D = (I = (U, A), d)$ in which $d \notin A$ is an attribute called the *decision*. In this case indiscernibility relations are divided into the set $\{Ind(B) : B \subseteq A\}$ and the set $Ind(d)$. Relations between pairs of the form: a class of $Ind(B)$, a class of $Ind(d)$, are called *decision rules*; they are called *classification rules* in case d is an assignments of objects in U into distinct categories.

Formally, decision rules are expressed in the language of *descriptor logic*, see, e.g., [2], [20]; a *descriptor* is a word of the form $(a = v)$, with semantics of the form $[(a = v)] = \{u \in U : a(u) = v\}$. From words, sentences (formulas) are built by means of sentential connectives, with semantics defined inductively on complexity of formulas,

$$\begin{array}{c} 1.\ [\alpha \vee \beta] = [\alpha] \cup [\beta]. \\ 2.\ [\alpha \wedge \beta] = [\alpha] \cap [\beta]. \\ 3.\ [\neg\alpha] = U \setminus [\alpha]. \\ 4.\ [\alpha \Rightarrow \beta] = [\neg\alpha \vee \beta]. \end{array} \tag{1}$$

In this language, a decision rule is a formula,

$$\bigwedge_{a \in B} (a = v_a) \Rightarrow (d = w), \tag{2}$$

where $B \subseteq A$ and v_a is a value of a.

Language of descriptors allows also for representing entities in information system universes by means of their *information sets*, cf., [20]: $Inf_A(u) = \{(a = a(u)) : a \in A\}$ is the information set of $u \in U$.

1.2 Mereology vs. Rough Sets

There are reasons for turning to mereological theory of concepts [11] when foundations of rough sets are considered. Rough set theory solves the problem of how to specify a rough concept with the idea of approximation: given a concept Y, there exist by completeness of the containment relation $\subseteq$, two definable concepts $\underline{Y}$ and $\overline{Y}$ such that $\underline{Y} \subseteq Y \subseteq \overline{Y}$, $\underline{Y}$ is the largest definable subset of Y and $\overline{Y}$ is the smallest definable superset of Y.

The following points deserve attention in the above presented scheme:
APR 1. Definable concepts are unions of atomic concepts: indiscernibility classes.
APR 2. Non–definable concepts are approached with definable ones by means of containment.

APR 1–2 are particular cases of general constructs of mereology: the union of sets is a particular example of the *class operator* and containment is a particular *ingredient relation*. It follows that viewing rough sets from the viewpoint of mereology, one obtains a general theory of which rough set theory formed in the naive set theory language is a particular case. The class operator is the main tool in the scheme of granulation presented here, see [23], [24], [29], [30], [32], [33], [34].

Basic notion of mereology is the notion of a part, see [11].

The relation π of being a part is a non–reflexive and transitive relation on entities, i.e.,

PT 1. $\pi(u, u)$ for no entity u.

PT 2. $\pi(u, v)$ and $\pi(v, w)$ imply $\pi(u, w)$.

An example of part relation is the proper containment relation $\subset$ on sets.

The part relation π induces a partial order relation *ing* of an *ingredient* [11],

$$v \ ing \ u \text{ iff } \pi(v, u) \text{ or } v = u. \tag{3}$$

Clearly, *ing* is reflexive, weakly–antisymmetric and transitive, i.e., it is a partial order. An example is the containment relation $\subseteq$ on sets induced by the part relation $\subset$.

The union of sets operator used in constructions of approximations, has its counterpart in the *class operator Cls* [11]; it is applied to any non–empty collection F of entities to produce the entity $ClsF$; the formal definition is given in terms of the ingredient relation: an entity X is the class $ClsF$ if and only if the two conditions are satisfied,
CL 1. $u \ ing \ X$ for each $u \in F$.
CL 2. $u \ ing \ X$ implies the existence of entities v, w with the properties:

i. $v \ ing \ u$;
ii. $v \ ing \ w$;
iii. $w \in F$.

Informally, $ClsF$ collects entities whose each part has a part in common with an entity in F.

An example of $ClsF$ in case π is $\subset$ and *ing* is $\subseteq$ for F a non–empty collection of sets, is $\bigcup F$, the union of F.

1.3 Rough Mereology and Rough Inclusions

Rough mereology is a theory which combines ideas of mereology and rough set theory [36], [35], [23], [33], [34]. Its particular upshot is a variety of similarity measures called *rough inclusions* whose aim is to replace indiscernibility relations in analysis of uncertainty.

Similarity relations in information systems should be proper entities, i.e., they should be invariant under indiscernibility. This implies that they should be defined in terms of descriptors. Basic notions of rough merology, relations called rough inclusions, fulfill this demand.

To introduce rough inclusions, and to give some motivation for their properties, we recall the idea of Henri Poincaré (1905) [21]. Poincaré considered a relation $\tau_\rho(x,y)$ which holds for a given metric ρ if and only if $\rho(x,y) < \delta$ for some small positive δ . The relation τ_ρ is a *tolerance relation*, i.e., it is reflexive and symmetric. Tolerance relations, cf., [64], are basic similarity relations.

We generalize the idea of Poincaré. We let

$$\mu_\rho(x,y,r) \text{ iff } \rho(x,y) \leq 1-r; \tag{4}$$

the predicate μ_ρ does satisfy a number of conditions which follow from properties of the metric ρ.

1. Rh 1. $\mu_\rho(x,y,1)$ if and only if $x=y$.
2. Rh 2. If $\mu_\rho(x,y,1)$ and $\mu_\rho(z,x,r)$ then $\mu_\rho(z,y,r)$.
3. Rh 3. If $\mu_\rho(x,y,r)$ and $s<r$ then $\mu_\rho(x,y,s)$.
4. Rh 4. If $\mu_\rho(x,y,r)$ and $\mu_\rho(y,z,s)$ then $\mu_\rho(x,z,L(r,s))$,

where $L(r,s) = max\{0, r+s-1\}$ is the well–known Łukasiewicz functor of many–valued logics, see, e.g., [9] or [22].

Properties Rh 1 – Rh3 are singled out by us as characteristic for *rough inclusions*; property Rh 4. which does reflect the triangle inequality for the metric ρ, is the *transitivity property* of rough inclusions of the form μ_ρ.

An abstract definition of a rough inclusion will refer to properties Rh 1 – Rh3 with an additional important factor, viz., property Rh 1 will in general refer to an ingredient relation of mereology; it is evident that for rough inclusions of the form μ_ρ, the associated ingredient relation is the identity = and the part relation is empty.

A *rough inclusion* $\mu_\pi(x,y,r)$, where $x,y \in U$ are individual objects in the universe U, $r \in [0,1]$, and π is a part relation of a chosen mereological description of concepts, does satisfy the following requirements,

$$\begin{gathered} RI1.\ \mu_\pi(x,y,1) \Leftrightarrow x\ ing_\pi\ y. \\ RI2.\ \mu_\pi(x,y,1)\text{and } \mu_\pi(z,x,r) \text{ imply } \mu_\pi(z,y,r). \\ RI3.\ \mu_\pi(x,y,r) \text{ and } \wedge s<r\text{imply } \mu_\pi(x,y,s). \end{gathered} \tag{5}$$

These requirements seem to be intuitively clear. RI 1 demands that the predicate μ_π is an extension to the relation ing_π of the underlying system of mereology; RI 2 does express monotonicity of μ_π and RI 3 assures the reading: “to degree at least r”.

Condition RI 1 states that on U an exact decomposition into parts π is given and that μ_π extends this exact scheme into an approximate one. The exact scheme is a skeleton along which approximate reasoning is carried out. This interpretation opens up a venue for various applications to problems in which establishing the rules for propagation of uncertainty is indispensable.

2 Granulation of Knowledge: The Abstract Definition of a Granule of Knowledge

The issue of granulation of knowledge as a problem on its own, has been posed by L.A.Zadeh [63], [62]. Granularity is imbued into fuzzy set theory as computations in that theory go along lines of membership functions whose inverse values form elementary granules. The issue of granulation has been a subject of intensive studies within rough set community, as witnessed by a number of papers, e.g.,[12], [13], [14], [15], [41], [48], [60], [61].

Rough set context offers a natural venue for granulation, and indiscernibility classes were recognized as *elementary granules* whereas their unions serve as *granules of knowledge*; these granules and their direct generalizations to various similarity classes were subject to a research, see, e.g.,[12]– [15] for the idea of a granule as a neighborhood, [60], [61].

Granulation of knowledge and applications to knowledge discovery in the realm of approximation spaces were studied, among others, in [48].

A study of granule systems was also initiated within rough mereology in [36], [37], [38] and further pursued in [23],[24], [29], [30], [31], [32], [33], [34], in order to find general properties of granules and develop schemes for applications in logics for approximate reasoning [26], [27] and in rough neural computing [25], [28].

2.1 Granules of Knowledge

An abstract definition of a granule of knowledge requires fixing a rough inclusion μ on a universal class U of entities endowed with a compatible part relation π so that conditions RI 1–RI 3 of (5) are fulfilled. The definition of a granule $g^\mu(v,r)$ of radius r about the entity $v \in U$, where r is a real number between 0 and 1, given in [24], [29], [30] is as follows,

$$u \in \Psi^\mu(v,r) \text{ iff } \mu(u,v,r), \tag{6}$$

and

$$g^\mu(v,r) \text{ is } Cls\Psi^\mu(v,r). \tag{7}$$

2.2 Granular Reflections of Decision Systems

The idea of a granular reflection of a decision system was posed in [29]: for a given decision system $D = (I = (U,A)),d)$, a rough inclusion μ, and $r \in [0,1]$, the new universe $U^G_{r,\mu}$ is given consisting of all granules $g^\mu(v,r)$ of the given radius r. A strategy $\mathcal{G}$ is applied in order to choose a covering $Cov^G_{r,\mu}$ of the universe U by granules from $U^G_{r,\mu}$,

$$Cov^G_{r,\mu} = \mathcal{G}(U^G_{r,\mu}). \tag{8}$$

A strategy $\mathcal{S}$ is applied in order to assign the value $a^*(g)$ for each attribute $a \in A$ to each granule $g \in Cov^G_{r,\mu}$,

$$a^*(g) = \mathcal{S}(\{a(u) : u \in g\}). \tag{9}$$

The object $o(g)$ defined by the condition,

$$Inf_{A^*}(o(g)) = \{(a^*, a^*(g)) : a \in A\}, \tag{10}$$

where $A^* = \{a^* : a \in A\}$, is said to be the *granular reflection of the granule g*.

The *$\mathcal{G}, \mathcal{S}$–granular reflection of the decision system $D = (I = (U, A)), d)$* is a pair

$$D^* = (I^* = (Cov^G_{r,\mu}, A^*)), d^*).$$

The heuristic principle that

objects, similar with respect to conditional attributes in the set A, should also reveal similar (i.e., close) decision values, and therefore, in particular, granular counterparts to decision systems should lead to classifiers satisfactorily close in quality to those induced from original decision systems,

was stated in [29], and borne out by simple hand examples. In this work we verify this hypothesis with real data sets.

3 Classifiers: Rough Set Methods

For a decision system $D = (I = (U, A)), d)$, classifiers are sets of rules (2). Induction of rules was a subject of research in rough set theory since its beginning. In most general terms, building a classifier consists in searching in the pool of descriptors for their conjuncts that describe sufficiently well decision classes. As distinguished in [51], there are three main kinds of classifiers searched for: *minimal*, i.e., consisting of minimum possible number of rules describing decision classes in the universe, *exhaustive*, i.e., consisting of all possible rules, *satisfactory*, i.e., containing rules tailored to a specific use. Classifiers are evaluated globally with respect to their ability to properly classify objects, usually by *error* which is the ratio of the number of correctly classified objects to the number of test objects, *total accuracy* being the ratio of the number of correctly classified cases to the number of recognized cases, and *total coverage*, i.e, the ratio of the number of recognized test cases to the number of test cases.

Minimum size algorithms include LEM2 algorithm by Grzymala–Busse, see, e.g., [7], [8] and covering algorithm in RSES package [42]; exhaustive algorithms include, e.g., LERS system due to Grzymala–Busse [6] and the standard exhaustive algorithm, see [42] for its publicly available version. Minimal in a sense are classifying systems based on discernibility matrices and Boolean reasoning according to Skowron, see, e.g., [45],[46], [3], [44]. Minimal consistent sets of rules were introduced in Skowron–Rauszer [43]; in [58] they were shown to coincide with rules induced on the basis of local reducts, see, e.g., [20] for the reduct notion. Further developments include dynamic rules, approximate rules, and relevant rules as described in Bazan [3] as well as local rules [3] effective in implementations of algorithms based on minimal consistent sets of rules. Rough set based classification algorithms, especially those implemented in the RSES system [42], were discussed extensively in [4]; [52] contains a discussion of rough set classifiers along with some attempt at analysis of granulation in the process of knowledge discovery.

An important class of methods for classifier induction are those based on similarity or analogy reasoning; most generally, this method of reasoning assigns to an object u the value of an attribute a from the knowledge of values of a on a set $N(u)$ of objects whose elements are selected on the basis of a similarity relation, usually but not always based on an appropriate metric.

An extensive study of algorithms based on similarity relations is [44]; as the main tool in inducing similarity relations, *templates*, i.e., propositional formulas built from *generalized descriptors* of the form $(a \in W_a)$ where W_a is a subset of the value set V_a as well as metrics extracted from data like the Manhattan, Hamming, Euclidean, are used. A number of similarity measures built from these basic forms were tested in [44].

A realization of analogy–based reasoning idea is the *k–nearest neighbors* (k-nn) method in which for a fixed number k, and a given test object u, the value $a(u)$ is assigned from values of a at k nearest to u objects in the training set. Finding nearest objects is based on some similarity measure among objects that in practice is a metric. Metrics to this end are built on the two basic metrics: the Manhattan metric for numerical values and the Hamming metric for nominal values; basic metrics are enhanced in many ways to produce a finer, better adapted metrics, e.g., VDM [49], and its modifications, IVDM, WVDM [56], DBDVM [57], and further these metrics are endowed with weighting attributes subject to optimization of weights, local metrics etc.; an experimental study of this topic is given by Wojna [57].

Our approach is also based on similarity yet it is a distinct one. Our sets $N(u)$ for $u \in U$, are formed as granules of the form $g^{\mu}(u, r)$ with μ, r fixed; for each such granule g, and each attribute $a \in A \cup \{d\}$, the factored value $a^*(g)$ is defined by (9). Contrary to the practice of using a metric that combines values of all attributes, in our approach, attributes are involved independently; similarity is driven by the rough inclusion μ.

As a result, each granule g does produce a new object $o(g)$, see (10), possibly not among real objects. We should observe, nevertheless, that one can neither prove nor disprove the existence of those objects in the real world. Due to this in a sense virtuality feature, those objects are well suited to the role of intermediary between the training and test worlds, as they disappear from the final result.

4 The Łukasiewicz Rough Inclusion and Granules

A basic rough inclusion applied in this work is induced by the Hamming distance in information/decision systems; as shown, e.g., in [31], it is also induced from the t–norm of Łukasiewicz and thus it is called here the Łukasiewicz rough inclusion, denoted μ_L.

The Hamming distance ρ_H on the universe U of an information/decision system I/D, reduced modulo cardinality $|A|$ of the set A is given by,

$$\rho_H(u,v) = \frac{|\{a \in A : a(u) \neq a(v)\}|}{|A|}, \tag{11}$$

and in consequence, by (4),

$$\mu_L(u,v,r) \text{ iff } \frac{|Inf_A(u) \varDelta Inf_A(v)|}{|A|} \leq 1 - r, \tag{12}$$

where $\varDelta(X,Y) = (X \setminus Y) \cup (Y \setminus X)$.

Equivalently,

$$\mu_L(u,v,r) \text{ iff } \frac{|Inf_A(u) \cap Inf_A(v)|}{|A|} \geq r. \tag{13}$$

The rough inclusion μ_L can be obtained in the general scheme of producing rough inclusions from Archimedean t–norms due to Polkowski, see [24],[29], [30], [31]. We recall it here for completeness' sake.

It is well–known, see [16], that Archimedean t–norms, i.e., t–norms T [9], [22] which satisfy the condition $T(x,x) < x$ for $x \in (0,1)$, admit a functional characterization, a very special case of the Kolmogorov theorem,

$$T(x,y) = g_T(f_T(x) + f_T(y)), \tag{14}$$

where the function $f_T : [0,1] \to R$ is continuous decreasing with $f_T(1) = 0$, and $g_T : R \to [0,1]$ is the pseudo–inverse to f_T, see [16] (a discussion may be also found in [22]).

In order to define the rough inclusion μ_T induced by an archimedean rough inclusion T, we let,

$$\mu_T(u,v,r) \Leftrightarrow g(\frac{|dis_A(u,v)|}{|A|}) \geq r. \tag{15}$$

where $dis_A(u,v) = \{a \in A : a(u) \neq a(v)\}$ and its complement $ind_A(u,v) = U \times U \setminus dis_I(u,v)$.

The Łukasiewicz t–norm $L(x,y) = max\{0, x + y - 1\}$ is Archimedean with $f_L(x) = 1 - x = g_L(x)$, see [16], [22], and thus the induced according to (15) rough inclusion μ_L is defined as,

$$\mu_L(u,v,r) \text{ iff } \frac{|ind_A(u,v)|}{|A|} \geq r, \tag{16}$$

which is (13).

As a granule $g^{\mu_L}(v, r)$ collects all "nearest" to v objects within the radius r, this approach is a far–reaching extension of methods based on selections of a bound number of closest objects like the k-nn method; interdependence of attributes, is taken care of by the rough inclusion μ_L. As it is based on dis_A – function, r–closed objects have an r–fraction of attribute values identical, hence, remaining values may be assumed to be also close, so the new granular object is representative for all objects in the granule.

The transitivity property Rh 4, sect. 1.3, implies the property of granules based on μ_L, see [24], [33], [34],

$$u \; ing \; g^{\mu_L}(v, r) \text{ iff } \mu_L(u, v, r), \tag{17}$$

for $u, v \in U$. On the basis of (17), the granule $g^{\mu_L}(v, r)$ can be represented as the set $\{v \in U : \mu_L(v, u, r)$.

5 The Setting of Experiments

In experiments with real data sets, we accept total accuracy and total coverage coefficients as quality measures in comparison of classifiers given in this work.

As stated above, our hypothesis is that the granular reflection D^* of a data set D at sufficiently large granulation radii r preserves knowledge encoded in D to a satisfactory degree so given an algorithm $\mathcal{A}$ for rule induction, classifiers obtained from the training set *D(trn)* and its granular counterpart $D^*(trn)$ should agree with a small error on the test set *D(tst)*.

To put our results into a formal perspective, let us consider an operator $\mathcal{O}$ acting on a family $\mathcal{D}$ of decision systems and having values in the interval $[0, 1]$ (e.g., accuracy, coverage or some other measure of the quality of a classifier) along with an operator $\mathcal{F}$ acting on decision systems in $\mathcal{D}$ with new decision systems as values; for $\varepsilon \in (0, 1)$, we will say that $\mathcal{F}$ is an $(\mathcal{O}, \varepsilon)$*–operator relative to a family* $\mathcal{D}$ *of decision systems* whenever $\mathcal{O}(\mathcal{F}(D)) \geq (1 - \varepsilon) \cdot \mathcal{O}(D)$ for each $D \in \mathcal{D}$.

The following data sets have been used in this part of experiments.

- Fisher's Iris data, see [53], [42];
- Lymphography database, see [53];
- Hearth disease data set (Cleveland data), see [53], [42];
- Breast cancer data set, see [53], [42];
- Primary tumor data set, see [53];
- Credit card application approval data set (Australian credit), see [54], [42];
- Diabetes data set, see [42];
- Pima Indians diabetes data set, see [53].

As representative and well–established algorithms for rule induction in public domain, we have selected

- the exhaustive algorithm ;
- the covering algorithm of RSES with p=.1 [42];
- LEM2 algorithm, with p=.5, see [7], [42].

Table 1. Comparison of algorithms on Australian credit data. 345 training objects, 345 test objects.

algorithm	*accuracy*	*coverage*	*rule number*
$covering(p = .1)$	0.670	0.783	589
$covering(p = .5)$	0.670	0.783	589
$covering(p = 1.0)$	0.670	0.783	589
exhaustive	0.835	1.0	5149
$LEM2(p = .1)$	0.810	0.061	6
$LEM2(p = .5)$	0.906	0.368	39
$LEM2(p = 1.0)$	0.869	0.643	126

Table 1 shows a comparison of these algorithms on the data set Australian credit split into the training and test sets with the ratio 1:1.

For any granule g and any attribute b in the set $A \cup d$ of attributes, the reduced attribute's $\overline{b}$ value at the granule g has been estimated by means of the majority voting strategy and ties have been resolved at random; majority voting is one of most popular strategies and was frequently applied within rough set theory, see, e.g., [44], [57].

We also use the simplest strategy for covering finding, i.e., we select coverings by ordering objects in the set U and choosing sequentially granules about them in order to obtain an irreducible covering; a random choice of granules is applied in sections in which this is specifically mentioned and the result is an essential covering, i.e., in the process of granule random selection, each subsequent granule contains objects not covered by already selected granules.

The only enhancement of the simple granulation is discussed in sect. 6 where the concept–dependent granules are considered; this approach yields even better classification results.

5.1 Results of Experiments: Training Table=Test Table

Here, we report results for the data sets chosen in case when the training sample is also the test sample. Quality of classification is measured by means of two parameters: the total accuracy and the total coverage.

Experiments have been carried out in accordance with the following procedure,

1. the data table (U, A) has been input;
2. classification rules have been found by means of each of the three algorithms;
3. classification of dataset objects in U has been found for each of the three classifications found at point 2;
4. given the granule radius, granules of that radius have been found;
5. a granular covering of the universe U has been chosen;
6. the corresponding granular decision system has been determined;
7. granular classifiers have been induced from the granular system in point 6 by means of each of algorithms in point 2;

8. classifications of objects in U have been found by means of each of classifiers in point 7;
9. classifications from points 3,8 have been compared with respect to adopted global measures of quality: total accuracy and total covering.

Results of each test are given in tables below. For this section, four data sets have been selected:

– Fisher's Iris data set;
– Lymphography data set;
– Hearth disease data set (Cleveland data);
– Breast cancer data set.

Results for these data sets are reported in following subsections. The radius value of *nil* indicates the non–granular case; the + sign means that the result in granular case is better than in non–granular case.

Iris data set. In Tables 2, 3, results of experiments with Iris data set are collected. Table 2 shows data about the size of samples, training and test, and

Table 2. Iris dataset:r=granule radius,tst=test sample size,trn=training sample size,rulcov=number of rules with covering algorithm,rulex=number of rules with exhaustive algorithm, rullem=number of rules with LEM2,acov=total accuracy with covering algorithm,ccov=total coverage with covering algorithm,aex=total accuracy with exhaustive algorithm,cex=total coverage with exhaustive algorithm,alem=total accuracy with LEM2, clem=total coverage with LEM2

r	*tst*	*trn*	*rulcov*	*rulex*	*rullem*	*acov*	*ccov*	*aex*	*cex*	*alem*	*clem*
nil	150	150	104	246	12	1.0	0.987	1.0	1.0	1.0	0.540
0.0	150	1	4	0	0	1.0	0.080	0.0	0.0	0.0	0.0
0.25	150	24	33	37	3	0.669	0.947	0.669	0.947	0.955	0.293
0.5	150	66	80	161	11	0.930	0.947	0.940	0.993	0.960	0.5
0.75	150	131	103	238	12	0.993	0.987	1.0	1.0	1.0	0.540
1.0	150	147	104	246	12	1.0	0.987	1.0	1.0	1.0	0.540

Table 3. Iris dataset:comparison; r=granule radius,acerr= abs.total accuracy error with covering algorithm,ccerr= abs.total coverage error with covering algorithm,aexerr=abs.total accuracy error with exhaustive algorithm,cexerr=abs.total coverage error with exhaustive algorithm,alemerr=abs.total accuracy error with LEM2, clemerr=abs.total coverage error with LEM2, sper=training sample size as fraction of the original size,rper= max granular rule set size as fraction of the original rule set size

r	*acerr*	*ccerr*	*aexerr*	*cexerr*	*alemerr*	*clemerr*	*sper*	*rper*
nil	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
0.0	0.0	0.907	1.0	1.0	1.0	0.540	0.006	0.038
0.25	0.331	0.04	0.331	0.053	0.045	0.247	0.16	0.317
0.5	0.07	0.04	0.06	0.007	0.04	0.04	0.44	0.916
0.75	0.007	0.0	0.0	0.0	0.0	0.0	0.873	1.0
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.980	1.0

number of rules as well as the results of classification, and Table 3 reports radii of granulation versus absolute errors in accuracy and coverage along with relative sizes of the training as well as the rule sets.

The comparison of classifiers for the Iris data induced by means of exhaustive, covering and LEM2 algorithms from the original data set with classifiers induced by means of these algorithms from granular counterparts for all possible radii, shows that, indeed, although a substantial reduction in size of both data set and decision rule set is witnessed, yet still a satisfactory quality of the classifier is maintained. Table 3 shows this synthetically.

Conclusions in case of Iris data set. Tables 2, 3 show that by $r = 0.5$, the reduction in size of the set of objects is 56 percent whereas the reduction in number of rules is from 23 percent for covering algorithm through 35 percent for exhaustive algorithm to 9 percent in case of LEM2, yet the classification error both for accuracy and coverage is not greater than 0.07 for all algorithms; for exhaustive algorithm, at $r = 0.5$, error in accuracy is 6 percent and in coverage .7 percent; at $r = 0.75$, the size reduction is 13 percent with almost the same number of rules and classification error is less than 0.007. Strikingly, at $r = 0.25$ in which case the reduction in size is 78 percent and reduction in rule number is between 85 percent for the exhaustive algorithm and 70 percent for the covering algorithm, the classification error drops below 0.31, i.e., by 31 percent for all algorithms in accuracy, and 0.05, i.e., by 5 percent in coverage for exhaustive as well as covering algorithms, maintaining about 50 percent drop in coverage with LEM2.

5.2 Lymphography Data Set

We record here results of experiments with Lymphography data set, under the same conditions. Tables 4, 5 show respective results.

Conclusions in case of Lymphography data set. From the radius of 0.8(3) on, the exhaustive algorithm trained on the granular system yields accuracy within 92 percent of the value in non–granular case with reduction in the training set size of 39.5 percent and reduction in the rule set size of 26 percent. Coverage in granular case lies within 98.6 percent of the value for the original data set with reduction in the training set of 94 percent and reduction in the rule set size of more than 98 percent (71 to 6794).

With the covering algorithm, coverage in the granular case exceeds or equals that of the original system at the radius of 0.38 in which case reduction in size is 96 percent and reduction in rule number is 91 percent whereas accuracy falls within 0.3 of the value 1.0 for the original system with the radius of 0.61 and reduction in size of 90.5 percent, and reduction in rule number of 90.5 percent. Error in accuracy of 0.1 is obtained at the radius of 0.8(3) in which case reduction in size is 39.5 percent and reduction in rule number is 40.9 percent.

In case of LEM2, from the radius of .722 on, accuracy error is less than .13 (13 percent) and error in coverage is less than .04 (.7 percent), with reduction in number of objects of 76 percent and reduction in size of the rule set of 69 percent.

Table 4. Lymphography dataset:r=granule radius,tst=test sample size,trn=training sample size,rulcov=number of rules with covering algorithm,rulex=number of rules with exhaustive algorithm, rullem=number of rules with LEM2,acov=total accuracy with covering algorithm,ccov=total coverage with covering algorithm,aex=total accuracy with exhaustive algorithm,cex=total coverage with exhaustive algorithm,alem=total accuracy with LEM2, clem=total coverage with LEM2

r	*tst*	*trn*	*rulcov*	*rulex*	*rullem*	*acov*	*ccov*	*aex*	*clex*	*alem*	*clem*
nil	148	148	115	6794	13	1.0	0.932	1.0	1.0	1.0	0.527
0.0	148	1	18	0	0	1.0	0.547	0.0	0.0	0.0	0.0
0.0555556	148	1	18	0	0	1.0	0.547	0.0	0.0	0.0	0.0
0.111111	148	1	18	0	0	1.0	0.547	0.0	0.0	0.0	0.0
0.166667	148	2	19	0	1	1.0	0.547	0.0	0.0	1.0	0.311
0.222222	148	2	19	0	1	1.0	0.547	0.0	0.0	1.0	0.311
0.277778	148	2	19	0	1	1.0	0.547	0.0	0.0	1.0	0.311
0.333333	148	3	21	0	1	1.0	0.547	0.0	0.0	1.0	0.311
0.388889	148	5	11	11	1	0.486	0.959	0.486	0.959	0.821	0.378
0.444444	148	5	9	18	2	0.442	0.932	0.553	0.953	0.0	0.0
0.5	148	8	23	71	2	0.603	0.986	0.534	0.986	0.913	0.155
0.555556	148	9	15	158	2	0.562	0.986	0.521	0.986	0.689	0.5
0.611111	148	14	12	265	2	0.720	0.892	0.671	0.986	0.781	0.216
0.666667	148	19	23	493	3	0.752	0.926	0.767	0.986	0.788	0.351
0.722222	148	35	26	1353	4	0.727	0.939	0.824	1.0	0.867	0.507
0.777778	148	56	45	2338	4	0.775	0.932	0.845	1.0	0.875	0.541
0.833333	148	91	68	5025	12	0.896	0.905	0.926	1.0	0.944	0.486
0.888889	148	130	117	6545	12	0.957	0.953	0.993	1.0	1.0	0.534
0.944444	148	145	115	6826	13	1.0	0.932	1.0	1.0	1.0	0.547

5.3 Heart Disease Data Set

The following Tables 6, 7 give results of experiments with Heart disease data set (Cleveland).

Conclusions for Heart disease data set. In case of these data, accuracy by the covering algorithm falls within 0.07 of the original value, and coverage is the same as with original data, from the radius of 0.692308 on, at which radius, object size reduction is 20.4 percent and rule set size reduction is 12 percent. Error in coverage is within 10.4 percent from the radius of 0.307692 on, at which radius object size reduction is 97.7 percent and rule set size reduction is 95.8 percent. Error in accuracy is within 27.5 percent from this radius on.

In case of exhaustive algorithm, accuracy falls within 0.27 (27 percent of the value with original data set), and coverage within 0.037 of values for original data set at the radius of 0.307692, where object size reduction is 97.8 percent and rule set size reduction is 99.5 percent. Accuracy falls within error of 11.5 percent of the original value from the radius of 0.538462 on, where reduction in object set size is 51.2 percent and reduction in rule set size is 58.3 percent; accuracy error is less than 3 percent from $r = 0.692$ on, with maximal coverage of 1.0, when reduction in object number is 20.4 percent and in rule size 18 percent.

Table 5. Lymphography dataset:comparison; r=granule radius,acerr= abs.total accuracy error with covering algorithm,ccerr= abs.total coverage error with covering algorithm,aexerr=abs.total accuracy error with exhaustive algorithm,cexerr=abs.total coverage error with exhaustive algorithm,alemerr=abs.total accuracy error with LEM2, clemerr=abs.total coverage error with LEM2, sper=training sample size as fraction of the original size,rper= max rule set size as fraction of the original size

r	*acerr*	*ccerr*	*aexerr*	*cexerr*	*alemerr*	*clemerr*	*sper*	*rper*
nil	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
0.0	0.0	0.385	1.0	1.0	1.0	0.527	0.006	0.156
0.055555	0.0	0.385	1.0	1.0	1.0	0.527	0.006	0.156
0.111111	0.0	0.385	1.0	1.0	1.0	0.527	0.006	0.156
0.166667	0.0	0.385	1.0	1.0	0.0	0.216	0.013	0.165
0.222222	0.0	0.385	1.0	1.0	0.0	0.216	0.013	0.165
0.277778	0.0	0.385	1.0	1.0	0.0	0.216	0.013	0.165
0.333333	0.0	0.385	1.0	1.0	0.0	0.216	0.020	0.182
0.388889	0.514	0.027+	0.514	0.041	0.179	0.149	0.033	0.095
0.444444	0.558	0.0	0.447	0.047	1.0	0.527	0.033	0.154
0.5	0.397	0.054+	0.466	0.014	0.087	0.372	0.054	0.2
0.555556	0.438	0.054+	0.479	0.014	0.311	0.027	0.06	0.154
0.611111	0.28	0.04	0.329	0.014	0.219	0.311	0.094	0.154
0.666667	0.248	0.006	0.233	0.014	0.212	0.176	0.128	0.23
0.722222	0.273	0.007+	0.176	0.0	0.133	0.02	0.236	0.308
0.777778	0.225	0.0	0.155	0.0	0.125	0.014+	0.378	0.391
0.833333	0.104	0.027	0.074	0.0	0.056	0.041	0.614	0.923
0.888889	0.043	0.021+	0.007	0.0	0.0	0.007+	0.878	1.017
0.944444	0.0	0.0	0	0	0.0	0.02+	0.979	1.005

LEM2 algorithm achieves with granular systems error in accuracy less than 0.115 (11.5 percent) and error in coverage less than 0.02 (0.4 percent) from the radius of 0.538462 on, with reduction in object size of 51 percent and reduction of rule set size of 78 percent.

Breast cancer data set. Tables 8, 9 give results of tests with Breast cancer data set.

Conclusions for Breast cancer data set. In case of covering algorithm, coverage is within error of 23 percent for all radii, and it does exceed coverage in case of the original data set from the radius of 0.(6) on, in which case reduction in object size is 84 percent, and reduction in rule set size is 94 percent. Accuracy is within 25 percent error for all radii, and it falls to this error bound for radii of 0.(6), 0.(7); for all other radii error is less than 11 percent.

For exhaustive algorithm, coverage is 1.0 for all radii recorded whereas accuracy keeps within 15 percent error bound for all radii recorded, reaching the best result of 10.5 percent at the radius of 0.(8) at reduction in object size of 38 percent and reduction in rule set size of 17 percent.

Table 6. Heart dataset:r=granule radius,tst=test sample size,trn=training sample size,rulcov=number of rules with covering algorithm,rulex=number of rules with exhaustive algorithm, rullem=number of rules with LEM2,acov=total accuracy with covering algorithm,ccov=total coverage with covering algorithm,aex=total accuracy with exhaustive algorithm,cex=total coverage with exhaustive algorithm,alem=total accuracy with LEM2, clem=total coverage with LEM2

r	*tst*	*trn*	*rulcov*	*rulex*	*rullem*	*acov*	*ccov*	*aex*	*cex*	*alem*	*clem*
nil	270	270	275	5352	42	1.0	0.993	1.0	1.0	1.0	0.504
0.0	270	1	13	0	0	1.0	0.556	0.0	0.0	0.0	0.0
0.0769231	270	1	13	0	0	1.0	0.556	0.0	0.0	0.0	0.0
0.153846	270	1	13	0	0	1.0	0.556	0.0	0.0	0.0	0.0
0.230769	270	4	18	0	1	1.0	0.556	0.0	0.0	1.0	0.307
0.307692	270	6	17	28	2	0.754	0.919	0.727	0.963	0.4	0.019
0.384615	270	14	29	75	2	0.770	0.948	0.770	0.996	0.855	0.204
0.461538	270	23	53	131	3	0.732	0.941	0.778	1.0	0.848	0.341
0.538462	270	132	163	2231	9	0.750	0.889	0.896	1.0	0.896	0.496
0.615385	270	132	171	2114	12	0.725	0.904	0.885	1.0	0.885	0.485
0.692308	270	215	242	4389	29	0.925	0.933	0.970	1.0	0.945	0.541
0.769231	270	262	265	5220	41	0.985	0.985	1.0	1.0	1.0	0.500
0.846154	270	270	275	5352	42	1.0	0.993	1.0	1.0	1.0	0.504
0.923077	270	270	275	5352	42	1.0	0.993	1.0	1.0	1.0	0.504

Table 7. Heart dataset:comparison; r=granule radius,acerr= abs.total accuracy error with covering algorithm,ccerr= abs.total coverage error with covering algorithm,aexerr=abs.total accuracy error with exhaustive algorithm,cexerr=abs.total coverage error with exhaustive algorithm,alemerr=abs.total accuracy error with LEM2, clemerr=abs.total coverage error with LEM2, sper=training sample size as fraction of the original size,rper= max rule set size as fraction of the original size

r	*acerr*	*ccerr*	*aexerr*	*cexerr*	*alemerr*	*clemerr*	*sper*	*rper*
nil	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
0.0	0.0	0.437	1.0	1.0	1.0	0.504	0.0037	0.047
0.0769231	0.0	0.437	1.0	1.0	1.0	0.504	0.0037	0.047
0.153846	0.0	0.437	1.0	1.0	1.0	0.504	0.0037	0.047
0.230769	0.0	0.437	1.0	1.0	0.0	0.197	0.0148	0.065
0.307692	0.246	0.074	0.273	0.037	0.6	0.485	0.022	0.062
0.384615	0.23	0.045	0.23	0.004	0.145	0.3	0.052	0.105
0.461538	0.268	0.052	0.222	0.0	0.152	0.163	0.085	0.193
0.538462	0.25	0.104	0.104	0.0	0.104	0.008	0.489	0.593
0.615385	0.275	0.089	0.115	0.0	0.115	0.019	0.489	0.622
0.692308	0.075	0.06	0.03	0.0	0.055	0.037+	0.796	0.88
0.769231	0.015	0.008	0.0	0.0	0.0	0.004	0.97	0.976
0.8461540	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
0.923077	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0

Table 8. Breast dataset:r=granule radius,tst=test sample size,trn=training sample size,rulcov=number of rules with covering algorithm,rulex=number of rules with exhaustive algorithm, rullem=number of rules with LEM2,acov=total accuracy with covering algorithm,ccov=total coverage with covering algorithm,aex=total accuracy with exhaustive algorithm,cex=total coverage with exhaustive algorithm,alem=total accuracy with LEM2, clem=total coverage with LEM2

r	*tst*	*trn*	*rulcov*	*rulex*	*rullem*	*acov*	*ccov*	*aex*	*cex*	*alem*	*clem*
nil	286	286	223	1665	58	0.977	0.906	0.979	1.0	0.986	0.500
0.0	286	1	9	0	0	1.0	0.699	0.0	0.0	0.0	0.0
0.111111	286	2	9	0	1	1.0	0.699	0.0	0.0	1.0	0.573
0.222222	286	3	10	0	1	1.0	0.699	0.0	0.0	1.0	0.573
0.333333	286	6	13	0	1	1.0	0.703	0.0	0.0	1.0	0.573
0.444444	286	10	15	0	1	1.0	0.703	0.0	0.0	1.0	0.573
0.555556	286	30	22	0	1	1.0	0.703	0.0	0.0	1.0	0.598
0.666667	286	46	14	104	2	0.733	0.983	0.752	1.0	0.791	0.654
0.777778	286	94	41	570	3	0.749	0.962	0.797	1.0	0.827	0.689
0.888889	286	176	146	1387	15	0.869	0.937	0.878	1.0	0.872	0.545
1.0	286	266	216	1627	60	0.977	0.927	0.979	1.0	0.973	0.524

Table 9. Breast dataset:comparison; r=granule radius,acerr= abs.total accuracy error with covering algorithm,ccerr= abs.total coverage error with covering algorithm,aexerr=abs.total accuracy error with exhaustive algorithm,cexerr=abs.total coverage error with exhaustive algorithm,alemerr=abs.total accuracy error with LEM2, clemerr=abs.total coverage error with LEM2, sper=training sample size as fraction of the original size,rper= max rule set size as fraction of the original size

r	*acerr*	*ccerr*	*aexerr*	*cexerr*	*alemerr*	*clemerr*	*sper*	*rper*
nil	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
0.0	0.023+	0.207	0.979	1.0	0.986	0.500	0.003	0.04
0.111111	0.023+	0.207	0.979	1.0	0.014+	0.073+	0.006	0.04
0.222222	0.023+	0.207	0.979	1.0	0.014+	0.073+	0.01	0.045
0.333333	0.023+	0.203	0.979	1.0	0.014+	0.073+	0.02	0.058
0.444444	0.023+	0.203	0.979	1.0	0.014+	0.073+	0.035	0.067
0.555556	0.023+	0.203	0.979	1.0	0.014+	0.098+	0.105	0.099
0.666667	0.244	0.077+	0.227	0.0	0.195	0.154+	0.16	0.062
0.777778	0.228	0.056+	0.182	0.0	0.159	0.189+	0.329	0.342
0.888889	0.108	0.031+	0.101	0.0	0.114	0.045+	0.615	0.833
1.0	0.0	0.021+	0.0	0.0	0.013	0.024+	0.93	1.034

LEM2 coverage is better for granular systems than for original one from the radius of 0.(1) on, where reduction in object size is 99.4 percent and reduction in rule set size is 98 percent. Accuracy with LEM2, keeps within error bound of 20 percent for all radii, and it obtains the best result of 10 percent error at the radius of 0.(8) with reduction in object size of 38 percent and reduction in rule size of 74 percent.

5.4 Results of Experiments with Train-and-Test in 1:1 Ratio

Train-and-test, is a method in which data set is split into the training and testing parts; we use the ratio of 1:1, i.e., rules are induced on 50 percent of objects and tested on the remaining 50 percent.

The procedure in this case have been as follows.

1. the data table (U, A) has been input;
2. classification rules have been found on the training subtable of 50 percent of objects by means of each of the three algorithms;
3. classification of dataset objects in the test subtable of remaining 50 percent of objects has been found for each of the three classifications found at point 2;
4. given the granule radius, granules of that radius have been found on the training subtable;
5. a granular covering of the training subtable has been chosen;
6. the corresponding granular decision system has been determined;
7. granular classifiers have been induced from the granular system in point 6 by means of each of algorithms in point 2;
8. classifications of objects in the test subtable have been found by means of each of classifiers in point 7;
9. classifications from points 3,8 have been compared with respect to adopted global measures of quality: total accuracy and total covering.

We report here results of experiments with Primary tumor data set, Australian credit data set, Diabetes data set.

5.5 Primary Tumor Data Set

Results are shown in Tables 10, 11.

Conclusions for Primary tumor data set. With covering algorithm, accuracy in granular case is always higher than with original training data set, coverage begins to be within error of 25 percent with the radius of 0.823529 where reduction in training object set size is 68.05 percent and reduction in rule set size is 83 percent; at $r = .941$, where reductions in object and rule sizes are , resp., 36.7 and 42.4 percent, coverage in granular case exceeds coverage in non–granular case.

For exhaustive algorithm,accuracy is better with granular than original training set from the radius of 0.647059 on where reduction in size of training set is 92.9 percent and reduction in size of rule set is almost 100 percent (11 versus 4186). Coverage falls within error bound of 22.3 percent from the radius of 0.823529 on, where reduction in training st size is 68.2 percent and reduction in size of rule set is 75.5 percent; it becomes the same as in non–granular case at $r = .941$ with reduction in object size of 36.7 percent.

LEM2 exceeds accuracy of classifier trained on original training table with accuracy of granular classifier from the radius of 0.705882 on where reduction in training set size is 89.95 percent and reduction in rule set size is 93 percent. Coverage for granular classifier is better or within error of 13.5 percent from the radius of 0.882353 where reduction in size of the training set is 55.6 percent and reduction in size of rule set is 60.5 percent.

Table 10. Primary tumor dataset:r=granule radius,tst=test sample size,trn=training sample size,rulcov=number of rules with covering algorithm,rulex=number of rules with exhaustive algorithm, rullem=number of rules with LEM2,acov=total accuracy with covering algorithm,ccov=total coverage with covering algorithm,aex=total accuracy with exhaustive algorithm,cex=total coverage with exhaustive algorithm,alem=total accuracy with LEM2, clem=total coverage with LEM2

r	*tst*	*trn*	*rulcov*	*rulex*	*rullem*	*acov*	*ccov*	*aex*	*cex*	*alem*	*clem*
nil	170	169	170	4186	43	0.268	0.900	0.253	0.976	0.5	0.259
0.0	170	1	17	0	0	1.0	0.247	0.0	0.0	0.0	0.0
0.0588235	170	1	17	0	0	1.0	0.247	0.0	0.0	0.0	0.0
0.117647	170	1	17	0	0	1.0	0.247	0.0	0.0	0.0	0.0
0.176471	170	1	17	0	0	1.0	0.247	0.0	0.0	0.0	0.0
0.235294	170	1	17	0	0	1.0	0.247	0.0	0.0	0.0	0.0
0.294118	170	1	17	0	0	1.0	0.247	0.0	0.0	0.0	0.0
0.352941	170	1	17	0	0	1.0	0.247	0.0	0.0	0.0	0.0
0.411765	170	2	17	0	1	1.0	0.247	0.0	0.0	1.0	0.188
0.470588	170	3	18	0	1	1.0	0.247	0.0	0.0	1.0	0.188
0.529412	170	5	18	0	1	1.0	0.247	0.0	0.0	1.0	0.188
0.588235	170	8	19	0	1	1.0	0.247	0.0	0.0	1.0	0.188
0.647059	170	12	7	11	1	0.611	0.318	0.547	0.376	0.0	0.0
0.705882	170	17	12	40	3	0.543	0.476	0.457	0.476	0.667	0.035
0.764706	170	33	12	108	4	0.437	0.512	0.468	0.553	0.769	0.076
0.823529	170	54	28	1026	11	0.422	0.682	0.434	0.759	0.586	0.171
0.882353	170	75	47	3640	17	0.417	0.776	0.308	0.859	0.579	0.224
0.941176	170	107	98	4428	24	0.329	0.929	0.295	0.976	0.466	0.341
1.0	170	151	140	4249	36	0.303	0.912	0.283	0.976	0.500	0.341

Australian credit data set. Tables 12, 13 present results obtained in case of Australian credit data set.

Conclusions for Australian credit data set. With covering algorithm, accuracy is better or within error of 1 percent for all radii, coverage is better or within error of 4.5 percent from the radius of 0.214860 on where training set size reduction is 99 percent and reduction in rule set size is 98 percent.

With exhaustive algorithm, accuracy is within error of 10 percent from the radius of 0.285714 on, and it is better or within error of 4 percent from the radius of 0.5 where reduction in training set size is 85 percent and reduction in rule set size is 95 percent. The result of .875 at $r = .714$ is among the best at all (see Table 1). Coverage is better from $r = .214$ in the granular case, reduction in objects is 99 percent, reduction in rule size is almost 100 percent.

LEM2 gives accuracy better or within 2.6 percent error from the radius of 0.5 where training set size reduction is 85 percent and rule set size reduction is 96 percent. Coverage is better or within error of 7.3 percent from the radius of .571429 on where reduction in training set size is 69.6 percent and rule set size is reduced by 96 percent.

Table 11. Primary tumor dataset:comparison; r=granule radius,acerr= abs.total accuracy error with covering algorithm,ccerr= abs.total coverage error with covering algorithm,aexerr=abs.total accuracy error with exhaustive algorithm,cexerr=abs.total coverage error with exhaustive algorithm,alemerr=abs.total accuracy error with LEM2, clemerr=abs.total coverage error with LEM2, sper=training sample size as fraction of the original size,rper= max rule set size as fraction of the original size

r	*acerr*	*ccerr*	*aexerr*	*cexerr*	*alemerr*	*clemerr*	*sper*	*rper*
nil	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
0.0	0.732+	0.653	0.253	0.976	0.500	0.259	0.0059	0.1
0.0588235	0.732+	0.653	0.253	0.976	0.500	0.259	0.0059	0.1
0.117647	0.732+	0.653	0.253	0.976	0.500	0.259	0.0059	0.1
0.176471	0.732+	0.653	0.253	0.976	0.500	0.259	0.0059	0.1
0.235294	0.732+	0.653	0.253	0.976	0.500	0.259	0.0059	0.1
0.294118	0.732+	0.653	0.253	0.976	0.500	0.259	0.0059	0.1
0.352941	0.732+	0.653	0.253	0.976	0.500	0.259	0.0059	0.1
0.411765	0.732+	0.653	0.253	0.976	0.500+	0.071	0.012	0.1
0.470588	0.732+	0.653	0.253	0.976	0.500+	0.071	0.018	0.106
0.529412	0.732+	0.653	0.253	0.976	0.500+	0.071	0.029	0.106
0.588235	0.732+	0.653	0.253	0.976	0.500+	0.071	0.047	0.112
0.647059	0.343+	0.582	0.294+	0.600	0.500	0.259	0.071	0.041
0.705882	0.275+	0.424	0.204+	0.500	0.167+	0.224	0.1	0.07
0.764706	0.169+	0.388	0.215+	0.423	0.269+	0.183	0.195	0.093
0.823529	0.154+	0.218	0.181+	0.217	0.086+	0.088	0.319	0.256
0.882353	0.149+	0.124	0.055+	0.117	0.079+	0.035	0.444	0.869
0.941176	0.061+	0.029+	0.042+	0.00	0.034	0.082+	0.633	1.058
1.0	0.035+	0.012+	0.030+	0.00	0.00	0.082+	0.893	1.015

5.6 Diabetes Data Set

Finally, we test granular approach with Diabetes data. Results are shown in Tables 14, 15.

Conclusions for Diabetes data set. With covering algorithm, coverage for granular systems induced classifiers is better or within error of 5.7 percent for all radii; accuracy is better for radii from .250 on where reduction in size of object set is 78 percent and reduction in size of rule set is 62 percent, with exception of the radius of .5 where error is less than 2 percent.

Exhaustive algorithm performs well for granular classifiers for all radii: both coverage and accuracy are better or within error of 3 percent (for coverage).

LEM2 yields results for both coverage and accuracy better with granular classifiers than with original system induced, for all radii.

Summing up, on diabetes data, granular classifiers perform better than the original one.

5.7 Effect of Granule Selection on Classification

In order to test the impact a choice of granular covering has had on classification, we have carried out experiments with the exhaustive algorithm, by selecting

Table 12. Australian credit dataset:r=granule radius, tst=test sample size, trn=training sample size, rulcov=number of rules with covering algorithm, rulex=number of rules with exhaustive algorithm, rullem=number of rules with LEM2, acov=total accuracy with covering algorithm, ccov=total coverage with covering algorithm, aex=total accuracy with exhaustive algorithm, cex=total coverage with exhaustive algorithm, alem=total accuracy with LEM2, clem=total coverage with LEM2

r	*tst*	*trn*	*rulcov*	*rulex*	*rullem*	*acov*	*ccov*	*aex*	*clex*	*alem*	*clem*
nil	345	345	571	5597	49	0.634	0.791	0.872	0.994	0.943	0.354
0.0	345	1	14	0	0	1.0	0.557	0.0	0.0	0.0	0.0
0.0714286	345	1	14	0	0	1.0	0.557	0.0	0.0	0.0	0.0
0.142857	345	2	16	0	1	1.0	0.557	0.0	0.0	1.0	0.383
0.214286	345	3	7	7	1	0.641	1.0	0.641	1.0	0.600	0.014
0.285714	345	4	10	10	1	0.812	1.0	0.812	1.0	0.0	0.0
0.357143	345	8	18	23	2	0.820	1.0	0.786	1.0	0.805	0.252
0.428571	345	20	29	96	2	0.779	0.826	0.791	1.0	0.913	0.301
0.5	345	51	88	293	2	0.825	0.843	0.838	1.0	0.719	0.093
0.571429	345	105	230	933	2	0.835	0.930	0.855	1.0	0.918	0.777
0.642857	345	205	427	3157	20	0.686	0.757	0.867	1.0	0.929	0.449
0.714286	345	309	536	5271	45	0.629	0.774	0.875	1.0	0.938	0.328
0.785714	345	340	569	5563	48	0.629	0.797	0.870	1.0	0.951	0.357
0.857143	345	340	570	5574	48	0.626	0.791	0.864	1.0	0.951	0.357
0.928571	345	342	570	5595	48	0.628	0.794	0.867	1.0	0.951	0.357
1.0	345	345	571	5597	49	0.634	0.791	0.872	0.994	0.943	0.354

Table 13. Australian credit dataset:comparison; r=granule radius,acerr= abs.total accuracy error with covering algorithm,ccerr= abs.total coverage error with covering algorithm,aexerr=abs.total accuracy error with exhaustive algorithm,cexerr=abs.total coverage error with exhaustive algorithm,alemerr=abs.total accuracy error with LEM2, clemerr=abs.total coverage error with LEM2, sper=training sample size as fraction of the original size,rper= max rule set size as fraction of the original size

r	*acerr*	*ccerr*	*aexerr*	*cexerr*	*alemerr*	*clemerr*	*sper*	*rper*
nil	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
0.0	0.366+	0.234	0.872	0.994	0.943	0.354	0.003	0.024
0.0714286	0.366+	0.234	0.872	0.994	0.943	0.354	0.003	0.024
0.142857	0.366+	0.234	0.872	0.994	0.057+	0.029+	0.0058	0.028
0.214286	0.007+	0.209+	0.231	0.006+	0.343	0.340	0.009	0.02
0.285714	0.178+	0.209+	0.06	0.006+	0.943	0.354	0.012	0.02
0.357143	0.186+	0.209+	0.086	0.006+	0.138	0.102	0.023	0.04
0.428571	0.145+	0.035+	0.081	0.006+	0.03	0.053	0.058	0.05
0.5	0.191+	0.052+	0.034	0.006+	0.224	0.261	0.148	0.154
0.571429	0.201+	0.139+	0.017	0.006+	0.025	0.423+	0.304	0.403
0.642857	0.052+	0.034	0.005	0.006+	0.014	0.095+	0.594	0.748
0.714286	0.005	0.017	0.003+	0.006+	0.005	0.026	0.896	0.942
0.785714	0.005	0.006+	0.002	0.006+	0.008+	0.003+	0.985	0.994
0.857143	0.008	0.0	0.008	0.006+	0.008+	0.003+	0.985	0.998
0.928571	0.006	0.003+	0.005	0.006+	0.008+	0.003+	0.991	0.999
1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0

Table 14. Diabetes dataset:r=granule radius,tst=test sample size,trn=training sample size,rulcov=number of rules with covering algorithm,rulex=number of rules with exhaustive algorithm, rullem=number of rules with LEM2,acov=total accuracy with covering algorithm,ccov=total coverage with covering algorithm,aex=total accuracy with exhaustive algorithm,cex=total coverage with exhaustive algorithm,alem=total accuracy with LEM2, clem=total coverage with LEM2

r	*tst*	*trn*	*rulcov*	*rulex*	*rullem*	*acov*	*ccov*	*aex*	*clex*	*alem*	*clem*
nil	384	384	627	3664	109	0.664	0.859	0.605	0.995	0.625	0.188
0.0	384	1	8	0	0	1.0	0.443	0.0	0.0	0.0	0.0
0.125	384	19	64	0	1	1.0	0.617	0.0	0.0	1.0	0.305
0.250	384	86	239	366	11	0.626	0.969	0.632	0.990	0.692	0.203
0.375	384	220	508	1897	44	0.655	0.935	0.593	0.997	0.700	0.312
0.5	384	332	585	3189	82	0.667	0.844	0.585	0.997	0.659	0.221
0.625	384	381	626	3654	111	0.661	0.859	0.597	0.995	0.625	0.188
0.750	384	384	627	3664	109	0.664	0.859	0.605	0.995	0.625	0.188
0.875	384	384	627	3664	109	0.664	0.859	0.605	0.995	0.625	0.188
1.0	384	384	627	3664	109	0.664	0.859	0.605	0.995	0.625	0.188

Table 15. Diabetes dataset:comparison; r=granule radius,acerr= abs.total accuracy error with covering algorithm,ccerr= abs.total coverage error with covering algorithm,aexerr=abs.total accuracy error with exhaustive algorithm,cexerr=abs.total coverage error with exhaustive algorithm,alemerr=abs.total accuracy error with LEM2, clemerr=abs.total coverage error with LEM2, sper=training sample size as fraction of the original size,rper= max rule set size as fraction of the original size

r	*acerr*	*ccerr*	*aexerr*	*cexerr*	*alemerr*	*clemerr*	*sper*	*rper*
nil	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
0.0	0.336+	0.416	0.605	0.995	0.625	0.188	0.0026	0.012
0.125	0.336+	0.242	0.605	0.995	0.375+	0.117+	0.0495	0.102
0.250	0.038	0.11+	0.027+	0.005	0.067+	0.015+	0.224	0.381
0.375	0.009	0.076+	0.012	0.002+	0.075+	0.124+	0.573	0.81
0.5	0.003+	0.015	0.02	0.002+	0.034+	0.033+	0.864	0.933
0.625	0.003	0.0	0.008	0.0	0.0	0.0	0.992	1.01
0.750	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
0.875	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0

randomly granular coverings from all groups of granular coverings with same cardinalities on the Heart data set. Here are the results, shown in Table 16. There are shown intervals in which obtained values belong.

Conclusions for granule selection. For Heart disease data set, total accuracy was found to be 0.807, and total coverage 1.0 with exhaustive algorithm on the full table. This values are achieved here with radius of at least 0.538462, and beginning with the radius of 0.384615, the error in total accuracy is at most 0.07, and the error in total coverage is at most 0.007. This shows robustness of granular approach with respect to random coverings with granules.

Table 16. Effect of covering choice on classification

radius	*accuracy*	*coverage*	*number.coverings*
0.0	0.0	0.0	1
0.0769231	0.0	0.0	3
0.153846	0.0	0.0	4
0.230769	0.0 – 0.789	0.0 – 1.0	6
0.307692	0.0 – 0.793	0.0 – 1.0	10
0.384615	0.737 – 0.799	0.967 – 1.00	13
0.461538	0.778 – 0.822	0.996 – 1.0	15
0.538462	0.881 – 0.911	1.0	22
0.615385	0.874 – 0.907	1.0	23
0.692308	0.963 – 0.974	1.0	13
0.769231	1.0	1.0	3
0.846154	1.0	1.0	1
0.923077	1.0	1.0	1
1.0	1.0	1.0	1

5.8 Experiments: Cross-Validation CV-10

We have carried also tests with 10-fold cross-validation [10], [5] for Pima Indians diabetes data with exhaustive and LEM2 algorithms, and random choice of coverings; results are reported in Tables 17, 18.

Conclusions for CV-10. For exhaustive algorithm, accuracy in granular case exceeds or equals that in non–granular case from the radius of .625 with slightly smaller sizes of training as well as rule sets and it reaches 95.2 percent of accuracy in non–granular case, from the radius of .25 with reductions in size of the training set of 82.6 percent and in the rule set size of 94 percent. The difference in coverage is less than .4 percent from $r = .25$ on, where reduction in training set size is 82.6 percent, and coverage in both cases is the same from the radius of .375 on with reductions in size of both training and rule set of 48, resp., 53 percent.

For LEM2, accuracy in both cases differs by less than 1 percent from $r = .25$ on, and it is better in granular case from $r = .125$ on; coverage is better in granular case from $r = .375$ on.

Table 17. 10-fold CV; Pima; exhaustive algorithm. r=radius, macc=mean accuracy, mcov=mean coverage, mrules=mean rule number, mtrn=mean size of training set.

r	*macc*	*mcov*	*mrules*	*mtrn*
nil	0.6864	0.9987	7629	692
0.125	0.0618	0.0895	5.9	22.5
0.250	0.6627	0.9948	450.1	120.6
0.375	0.6536	0.9987	3593.6	358.7
0.500	0.6645	1.0	6517.6	579.4
0.625	0.6877	0.9987	7583.6	683.1
0.750	0.6864	0.9987	7629.2	692
0.875	0.6864	0.9987	7629.2	692

Table 18. 10-fold CV; Pima; LEM2 algorithm. r=radius,macc=mean accuracy,mcov=mean coverage,mrules=mean rule number, mtrn=mean size of training set.

r	macc	mcov	mrules	mtrn
nil	0.7054	0.1644	227.0	692
0.125	0.900	0.2172	1.0	22.5
0.250	0.7001	0.1250	12.0	120.6
0.375	0.6884	0.2935	74.7	358.7
0.500	0.7334	0.1856	176.1	579.4
0.625	0.7093	0.1711	223.1	683.1
0.750	0.7071	0.1671	225.9	692
0.875	0.7213	0.1712	227.8	692

6 Concept–Dependent Granulation

A modification of the approach presented in results shown above is the *concept dependent* granulation; a *concept* in the narrow sense is a decision/classification class, cf., e.g., [7]. Granulation in this sense consists in computing granules for objects in the universe U and for all distinct granulation radii as previously, with the only restriction that given any object $u \in U$ and $r \in [0,1]$, the new concept dependent granule $g^{cd}(u,r)$ is computed with taking into account only objects $v \in U$ with $d(v) = d(u)$, i.e., $g^{cd}(u,r) = g(u,r) \cap \{v \in U : d(v) = d(u)\}$. This method increases the number of granules in coverings but it is also expected to increase quality of classification, as expressed by accuracy and coverage.

We show that this is the case indeed, by including results of the test in which exhaustive algorithm and random choice of coverings were applied tenfold to Australian credit data set, once with the "standard" by now granular approach and then with the concept dependent approach. The averaged results are shown in Table 19.

For comparison, we include the best results on Australian credit data set obtained by rough set based methods in Table 20.

Let us observe that in case of Australian credit data set (Table 20), best results are obtained with simple, descriptor–based approaches, not augmented by specific metrics (viz., *general templates*:accuracy of .886; *simple templates*: accuracy of .929; *concept dependent granular system*: accuracy of .9970).

Conclusions for concept dependent granulation. Concept dependent granulation, as expected, involves a greater number of granules in a covering, hence, a greater number of rules, which is perceptible clearly up to the radius of .714286 and for greater radii the difference is negligible. Accuracy in case of concept dependent granulation is always better than in the standard case, the difference becomes negligible at the radius of .857143 when granules become almost single indiscernibility classes. Coverage in concept dependent case is almost the same as in the standard case, the difference between the two not greater than .15

Table 19. Standard and concept dependent granular systems for Australian credit data set; exhaustive RSES algorithm:r=granule radius, macc=mean accuracy, mcov=mean coverage, mrules=mean number of rules, mtrn=mean training sample size; in each column first value is for standard, second for concept dependent

r	*macc*	*mcov*	*mrules*	*mtrn*
nil	1.0; 1.0	1.0; 1.0	12025; 12025	690; 690
0.0	0.0; 0.8068	0.0; 1.0	0; 8	1; 2
0.0714286	0.0; 0.7959	0.0; 1.0	0; 8.2	1.2; 2.4
0.142857	0.0; 0.8067	0.0; 1.0	0; 8.9	2.4; 3.6
0.214286	0.1409; 0.8151	0.2; 1.0	1.3; 11.4	2.6; 5.8
0.285714	0.7049; 0.8353	0.9; 1.0	8.1; 14.8	5.2; 9.6
0.357143	0.7872; 0.8297	1.0; 0.9848	22.6; 32.9	10.1; 17
0.428571	0.8099; 0.8512	1.0; 0.9986	79.6; 134	22.9; 35.4
0.5	0.8319; 0.8466	1.0; 0.9984	407.6; 598.7	59.7; 77.1
0.571429	0.8607; 0.8865	0.9999; 0.9997	1541.6; 2024.4	149.8; 175.5
0.642857	0.8988; 0.9466	1.0; 0.9998	5462.5; 6255.2	345.7; 374.9
0.714286	0.9641; 0.9880	1.0; 0.9988	9956.4; 10344.0	554.1; 572.5
0.785714	0.9900; 0.9970	1.0; 0.9995	11755.5; 11802.7	662.7; 665.7
0.857143	0.9940; 0.9970	1.0; 0.9985	11992.7; 11990.2	682; 683
0.928571	0.9970; 1.0	1.0; 0.9993	12023.5; 12002.4	684; 685
1.0	1.0; 1.0	1.0; 1.0	12025.0; 12025.0	690; 690

Table 20. Best results for Australian credit by some rough set based algorithms; in case $*$, reduction in object size is 49.9 percent, reduction in rule number is 54.6 percent; in case $**$, resp., 19.7, 18.2; in case $***$, resp., 3.6, 1.9

source	*method*	*accuracy*	*coverage*
[3]	SNAPM(0.9)	error=0.130	-
[44]	simple.templates	0.929	0.623
[44]	general.templates	0.886	0.905
[44]	closest.simple.templates	0.821	1.0
[44]	closest.gen.templates	0.855	1.0
[44]	tolerance.simple.templ.	0.842	1.0
[44]	tolerance.gen.templ.	0.875	1.0
[59]	adaptive.classifier	0.863	-
this.work	granular*.r=0.642857	0.8990	1.0
this.work	granular**.r=0.714826	0.964	1.0
this.work	granular***.concept.dependent.r=0.785714	0.9970	0.9995

percent from the radius of .428571, where the average number of granules in coverings is 5 percent of the number of objects. Accuracy at that radius is better by .04 i.e. by about 5 percent in the concept dependent case.

It follows that concept dependent granulation yields better accuracy whereas coverage is the same as in the standard case.

7 Experiments with Rough Inclusions from Residua of t–Norms and Extensions of μ_L

In this part, we report results of experiments with real data in which granulation has been performed according to rough inclusions induced from residual implications of t–norms. In these experiments, a subset of Australian credit data set has been used, with random choice of granular coverings and majority voting as the attribute factoring strategy.

First, an extension of the Łukasiewicz rough inclusion μ_L is proposed which depends on a chosen metric ρ bounded by 1 in the attribute value space V of (we assume for simplicity that ρ is suitable for all attributes). In experiments, ρ has been chosen as the Euclidean 1–metric $\rho(x, y) = min\{1, |x - y|\}$.

Then, given an $\varepsilon \in [0, 1]$, we let

$$\mu^{\varepsilon}_{\rho}(v, u, r)$$

if and only if

$$|\{a \in A : \rho(a(v), a(u)) < \varepsilon\}| \geq r \cdot |A|.$$

It is manifest that μ^{ε} is a rough inclusion if ρ is a non–archimedean metric, i.e., $\rho(u, w) \leq max\{\rho(u, v), \rho(v, w)\}$; otherwise the monotonicity condition RI2 of (5) need not be satisfied and this takes place with most popular metrics like Euclidean, Manhattan etc.

In this case, a rough inclusion μ^{*}_{ρ} defined as follows: $\mu^{*}_{\rho}(v, u, r)$ if and only if there exists an ε such that $\mu^{\varepsilon}_{\rho}(v, u, r)$. Then it is easy to check that μ^{*} is a rough inclusion. The parameter r is called the *catch radius*.

Granules induced by the rough inclusion μ^{*}_{ρ} with $r = 1$ have a simple structure: a granule $g^{\varepsilon}_{\rho}(u, 1)$ consists of all $v \in U$ such that $\rho(a(u), a(v)) \leq \varepsilon$.

The idea poses itself to use granules defined in this way to assign a decision class to an object u in the test set.

First, given an object u in the test set, a granule $g^{\varepsilon}_{\rho}(u, 1)$ is formed in the training set.

Thus, $g^{\varepsilon}_{\rho}(u, 1) = \{v \in$ training set : $\rho(a(u), a(v)) \leq \varepsilon$ for each attribute $a \in A\}$.

Next, for each value c of a decision class, the following factor is computed,

$$\text{param(c)} = \frac{|g^{\varepsilon}_{\rho}(u, 1) \cap c}{\text{cardinality of c in the training set}}, \tag{18}$$

cf., [3], [4], for a discussion of various strategies of voting for decision values.

The class c_u assigned to u is decided by

$$param(c_u) = max_c param(c), \tag{19}$$

with random resolution of ties.

In computing granules, the parameter ε is normalized to the interval $[0,1]$ as follows: first, for each attribute $a \in A$, the value $train(a) = max_{training\ set}a - min_{training\ set}a$ is computed and the real line $(-\infty,+\infty)$ is contracted to the interval $[min_{training\ set}a, max_{training\ set}a]$ by the mapping f_a,

$$f_a(x) = \begin{cases} min_{training\ set}a \text{ in case } x \leq min_{training\ set}a \\ x \text{ in case } x \in [min_{training\ set}a, max_{training\ set}a] \\ max_{training\ set}a \text{ in case } x \geq max_{training\ set}a. \end{cases} \tag{20}$$

When the value $a(u)$ for a test object u is off the range $[min_{training\ set}a, max_{training\ set}a]$, it is replaced with the value $f_a(a(u))$ in the range. For an object v, or a rule r with the value $a(v)$, resp., $a(r)$ of a denoted $a(v,r)$, the parameter ε is computed as $\frac{|a(v,r)-f_a(a(u))|}{train(a)}$.

We show results of experiments with these rough inclusions. Our data set was a subset of Australian credit data in which training set had 100 objects from class 1 and 150 objects from class 0 (which approximately yields the distribution of classes in the whole data set). The test set had 100 objects, 50 from each class. The exhaustive classifier applied to this data set gave accuracy of 0.79 and coverage of 1.0.

7.1 Results of Tests with Granules of Training Objects According to $\mu_\rho^\varepsilon(v,u,1)$ Voting for Decision

In Fig. 1 results of classification are given in function of ε for accuracy as well as for coverage.

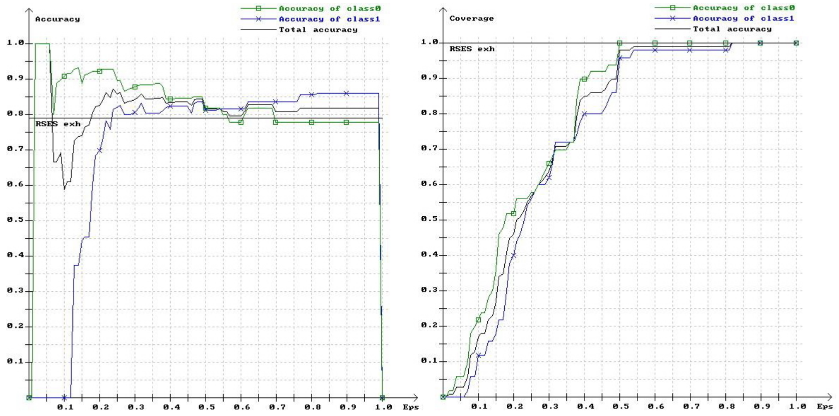


Fig. 1. Results for algorithm 1_v1, Best result for $\varepsilon = 0.62$: accuracy = 0.828283, coverage = 0.99

7.2 Results of Tests with Granules of Training Objects According to $\mu_\rho^\varepsilon(v, u, r)$

We return to the rough inclusion $\mu_\rho^*(v, u, r)$ with general radius r. The procedure applied in case of $\mu_\rho^\varepsilon(v, u, 1)$ can be repeated in the general setting. The resulting classifier is a function of two parameters ε, r. In Table 21 results are included where against values of the catch radius r the best value for ε's marked by the optimal value *optimal eps* is given for accuracy and coverage.

Table 21. (40%-60%)(1-0); Australian credit; Algorithm 1 v2. r catch=catch radius, optimal eps=Best ε, acc= accuracy, cov=coverage

r catch	*optimal eps*	*acc*	*cov*
nil	*nil*	0.79	1.0
0.071428	0	0.06	1.0
0.142857	0	0.66	1.0
0.214286	0.01	0.74	1.0
0.285714	0.02	0.83	1.0
0.357143	0.07	0.82	1.0
0.428571	0.05	0.82	1.0
0.500000	0	0.82	1.0
0.571429	0.08	0.84	1.0
0.642857	0.09	0.84	1.0
0.714286	0.16	0.85	1.0
0.785714	0.22	0.86	1.0
0.857143	0.39	0.84	1.0
0.928571	0.41	0.828283	0.99
1.000000	0.62	0.828283	0.99

7.3 Rough Inclusions and Their Weaker Variants Obtained from Residual Implications in Classification of Data

The residual implication of a continuous t–norm T is defined, see [9], or [22] as,

$$x \Rightarrow_T y \geq z \text{ if and only if } T(x, z) \leq y. \tag{21}$$

As shown in, e.g., [32], residual implications of continuous t–norms can supply rough inclusions according to a general formula,

$$\mu_\phi(v, u, r) \text{ iff } \phi(u) \Rightarrow_t \phi(v) \geq r, \tag{22}$$

where ϕ maps the set U of objects into $[0, 1]$ and $\phi(u) \leq \phi(v)$ if and only if $u\ ing\ v$ (ing is an ingredient relation of the underlying mereology).

Candidates for ϕ are proposed in [32], and a weak interesting variant of this class of rough inclusions is indicated. This variant uses sets $dis_\varepsilon(u, v) = \frac{|\{a \in A: \rho(a(u), a(v)) \geq \varepsilon\}|}{|A|}$, and $ind_\varepsilon(u, v) = \frac{|\{a \in A: \rho(a(u), a(v)) < \varepsilon\}|}{|A|}$, for $u, v \in U$, $\varepsilon \in [0, 1]$, where ρ is a metric $|x - y|$ on attribute value sets.

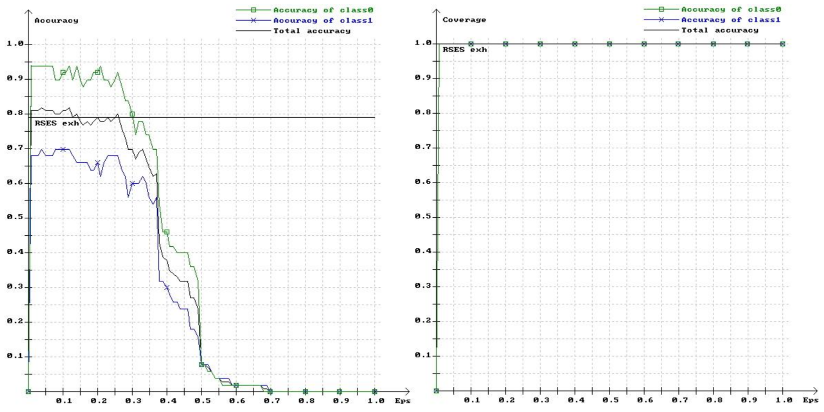


Fig. 2. Results for algorithm 5_v1, Best result for $\varepsilon = 0.04$, accuracy $= 0.82$, coverage $= 1$

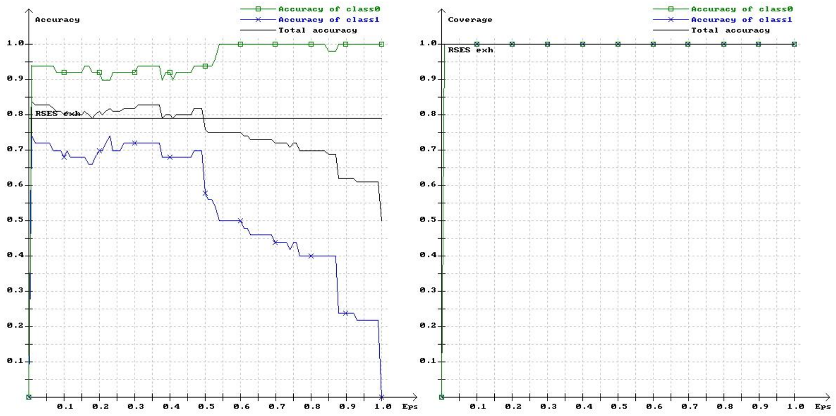


Fig. 3. Results for algorithm 5_v2, Best result for $\varepsilon = 0.01$, accuracy $= 0.84$, coverage $= 1$

The resulting weak variant of the rough inclusion μ_T is,

$$\mu_T(u, v, r) \text{ if and only if } dis_\varepsilon(u, v) \rightarrow_T \; ind_\varepsilon(u, v) \geq r. \tag{23}$$

Basic variants for three principal t–norms: the Łukasiewicz t–norm $L = max\{0, x + y - 1\}$, the product t–norm $P(x, y) = x \cdot y$, and $min\{x, y\}$ are, (the value in all variants is 1 if and only if $x \leq y$ so we give values only in the contrary case)

$$\mu_T(u, v, r) \text{ if and only if } \begin{cases} 1 - dis_\varepsilon(u, v) + ind_\varepsilon(u, v) \geq r \text{ for L} \\ \frac{ind_\varepsilon(u,v)}{dis_\varepsilon(u,v)} \geq r \text{ for P} \\ ind_\varepsilon(u, v) \geq r \text{ for min} \end{cases} \tag{24}$$

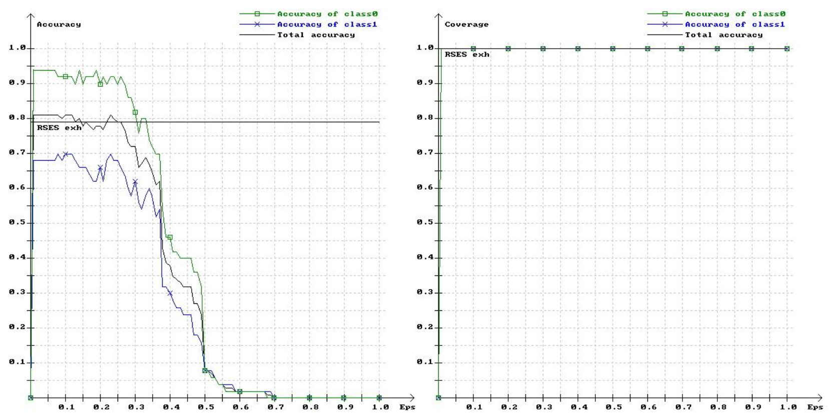


Fig. 4. Results for algorithm 6_v1, Best result for $\varepsilon = 0.01$, accuracy$=0.81$, coverage$=1$

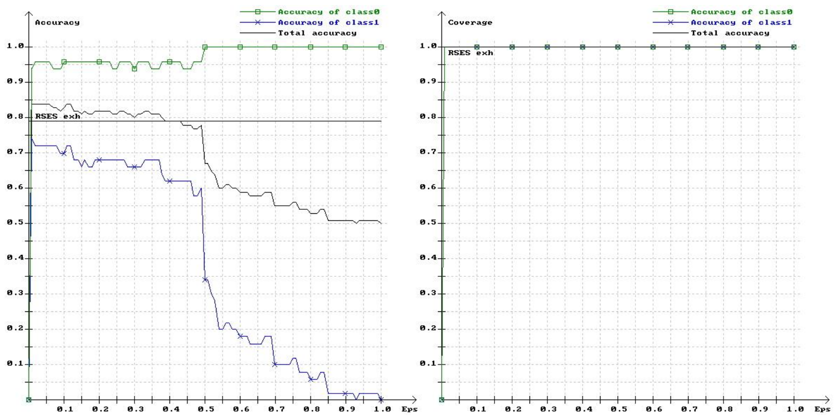


Fig. 5. Results for algorithm 6_v2, Best result for $varepsilon$ = 0.01, accuracy = 0.84, coverage = 1

Objects in the class c in the training set vote for decision at the test object u according to the formula: $p(c) = \frac{\sum_{v \in c} w(v,t)}{|c| \text{ in the training set}}$ where weight $w(v,t)$ is $dis_{\varepsilon}(u,v) \rightarrow_T \ ind_{\varepsilon}(u,v)$; rules induced from the training set pointing to the class c vote according to the formula $p(c) = \frac{\sum_r w(r,t) \cdot support(r)}{|c| \text{ in the training set}}$. In either case, the class c^* with $p(c*) = max_c p(c)$ is chosen. We include here results of tests with training objects and T=min (Fig.2)and rules and T=min (Fig.3).

Similarly, we include in Figs. 4, 5 results of tests with granules of training objects and rules for T=P, the product t–norm.

The results of tests in best cases for optimal values of ε exceed results obtained with the standard exhaustive algorithm.

8 Conclusions

It follows from experiments that granular decision systems offer substantial compression of data and classifier size, maintaining at the same time quality of classification on par with classifiers induced from original data.This confirms a hypothesis put forth in [29], [30] that granular decision systems preserve to a very high degree the information contained in the original decision system that is vital for classification tasks.

To validate the hypothesis statistically, we have collected results obtained in all tables for the first radius equal or greater than .5 along with the results for radius = *nil*, i.e., without granulation, matching them in pairs; the sample of $n = 21$ matched pairs has been subjected to the Wilcoxon signed rank test for matched pairs [55]. The hypothesis tested has been H_0:"the mean value in case of granular systems for accuracy/coverage is the same (or better) as 95 percent of the value of the mean in non–granular case", against the alternative H_a:"the mean value in case of granular systems for accuracy/coverage is less than 95 percent of the value of the mean in non–granular case" (i.e., one–tailed case). The p–value has been found of .4442 in case of accuracy and .2514 in case of coverage. Thus, there is no evidence on the basis of the sample to reject H_0.

Our results presented above justify the claim that *operators of random granular coverings for radii* $\geq .5$ *are* .05 $\mathcal{A}$*–operators for* $\mathcal{A}$ *– accuracy or coverage coefficients as yielded by exhaustive, covering or LEM2 algorithms.*

Voting by granules of various entities like training objects or rules obtained either from the training set or from the granulated reflection of the data set, proves a very effective classifier, better than the standard exhaustive classifying algorithm.

The results show the potential of granular classifying scheme, also in many possible applications. The more detailed study of voting strategies on basis of granular structures is given by the authors in [1].

Acknowledgment. Authors dedicate this work as homage to the late Professor Zdzisław Pawlak. The first author remembers with gratitude the support to his work offered kindly by Professor Zdzisław Pawlak and he recollects the atmosphere of honest work and regard for values created by Him.

References

1. Polkowski, L., Artiemjew, P.: On classifying mappings induced by granular structures. In this volume (submitted)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge U. Press, Cambridge (2004)
3. Bazan, J.G.: A comparison of dynamic and non–dynamic rough set methods for extracting laws from decision tables. In: Polkowski, L., Skowron, A. (eds.) Rough Sets in Knowledge Discovery, vol. 1, pp. 321–365. Physica Verlag, Heidelberg (1998)

4. Bazan, J.G., Son, N.H., Hoa, N.S., Synak, P., Wróblewski, J.: Rough set algorithms in classification problems. In: Polkowski, L., Tsumoto, S., Lin, T.Y. (eds.) Rough Set Methods and Applications, pp. 49–88. Physica Verlag, Heidelberg (2000)
5. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. John Wiley and Sons, New York (2001)
6. Grzymala–Busse, J.W.: LERS – a system for learning from examples based on rough sets. In: Słowiński, R. (ed.) Intelligent Decision Support: Handbook of Advances and Applications of the Rough Sets Theory, pp. 3–18. Kluwer, Dordrecht (1992)
7. Grzymala–Busse, J.W.: Data with missing attribute values: Generalization of rule indiscernibility relation and rule induction. In: Peters, J.F., Skowron, A., Grzymała-Busse, J.W., Kostek, B.z., Świniarski, R.W., Szczuka, M.S. (eds.) Transactions on Rough Sets I. LNCS, vol. 3100, pp. 78–95. Springer, Heidelberg (2004)
8. Grzymała-Busse, J.W., Hu, M.: A comparison of several approaches to missing attribute values in Data Mining. In: Ziarko, W.P., Yao, Y. (eds.) RSCTC 2000. LNCS (LNAI), vol. 2005, pp. 378–385. Springer, Heidelberg (2001)
9. Hájek, P.: Metamathematics of Fuzzy Logic. Kluwer, Dordrecht (1998)
10. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer, New York (2003)
11. Leśniewski, S.: Podstawy ogólnej teoryi mnogosci (On the foundations of set theory, in Polish). The Polish Scientific Circle, Moscow (1916) see also Topoi 2, 7–52 (1982)
12. Lin, T.Y.: Neighborhood systems and approximation in Database and Knowledge Based Systems. In: Proceedings of the 4th International Symposium on Methodologies of Intelligent Systems, Poster Session, pp. 75–86 (1989) [download]
13. Lin, T.Y.: Topological and fuzzy rough sets. In: Słowiński, R. (ed.) Intelligent Decision Support-Handbook of Applications and Advances of the Rough Sets Theory, pp. 287–304. Kluwer Academic Publishers, Dordrecht (1992)
14. Lin, T.Y.: From rough sets and neighborhood systems to information granulation and computing with words. In: Proceedings of the European Congress on Intelligent Techniques and Soft Computing, pp. 1602–1606. Verlag Mainz, Aachen (1997)
15. Lin, T.Y.: Granular computing: Examples, Intuitions, and Modeling. In: [39], pp. 40–44
16. Ling, C.-H.: Representation of associative functions. Publ. Math. Debrecen 12, 189–212 (1965)
17. Łukasiewicz, J.: Jan Łukasiewicz. Selected Works. North Holland and Polish Scientific Publishers, Amsterdam (1970)
18. Pal, S.K., Polkowski, L., Skowron, A. (eds.): Rough – Neural Computing. Techniques for Computing with Words. Springer, Berlin (2004)
19. Pawlak, Z.: Rough sets. Int. J. Computer and Information Sci. 11, 341–356 (1982)
20. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning about Data. Kluwer, Dordrecht (1991)
21. Poincare, H.: Science and Hypothesis. Walter Scott Publ., London (1905)
22. Polkowski, L.: Rough Sets. Mathematical Foundations. Physica Verlag, Heidelberg (2002)
23. Polkowski, L.: A rough set paradigm for unifying rough set theory and fuzzy set theory (a plenary lecture). In: Wang, G., Liu, Q., Yao, Y., Skowron, A. (eds.) RSFDGrC 2003. LNCS (LNAI), vol. 2639, pp. 70–78. Springer, Heidelberg (2003); also Fundamenta Informaticae 54, 67–88 (2003)
24. Polkowski, L.: Toward rough set foundations.Mereological approach (a plenary lecture). In: Tsumoto, S., Słowiński, R., Komorowski, J., Grzymała-Busse, J.W. (eds.) RSCTC 2004. LNCS (LNAI), vol. 3066, pp. 8–25. Springer, Heidelberg (2004)

25. Polkowski, L.: A rough–neural computation model based on rough mereology. In: [18], pp. 85–108
26. Polkowski, L.: A note on 3–valued rough logic accepting decision rules. Fundamenta Informaticae 61, 37–45 (2004)
27. Polkowski, L., Semeniuk–Polkowska, M.: On rough set logics based on similarity relations. Fundamenta Informaticae 64, 379–390 (2005)
28. Polkowski, L.: Rough–fuzzy–neurocomputing based on rough mereological calculus of granules. Intern. J. Hybrid Intell. Systems 2, 91–108 (2005)
29. Polkowski, L.: Formal granular calculi based on rough inclusions (a feature talk). In: [39], pp. 57–62
30. Polkowski, L.: A model of granular computing with applications (a feature talk). In: [40], pp. 9–16
31. Polkowski, L.: The paradigm of granular rough computing. In: Proceedings ICCI 2007. 6th IEEE Intern. Conf. on Cognitive Informatics, Lake Tahoe NV, USA, August 2007, pp. 145–163. IEEE Computer Society, Los Alamitos (2007)
32. Polkowski, L.: Granulation of knowledge in decision systems: The approach based on rough inclusions. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 271–279. Springer, Heidelberg (2007)
33. Polkowski, L.: A Unified approach to granulation of knowledge and granular computing based on rough mereology: A Survey. In: Pedrycz, W., Skowron, A., Kreinovich, V. (eds.) Handbook of Granular Computing, ch. 16. John Wiley, New York (2008)
34. Polkowski, L.: On the idea of using granular rough mereological structures in classification of data. In: Wang, G., Li, T., Grzymala-Busse, J.W., Miao, D., Skowron, A., Yao, Y. (eds.) RSKT 2008. LNCS (LNAI), vol. 5009, pp. 213–220. Springer, Heidelberg (2008)
35. Polkowski, L., Skowron, A.: Rough mereology: a new paradigm for approximate reasoning. International Journal of Approximate Reasoning 15(4), 333–365 (1997)
36. Polkowski, L., Skowron, A.: Rough mereological calculi of granules: A rough set approach to computation. Computational Intelligence. An International Journal 17(3), 472–492 (2001)
37. Polkowski, L., Skowron, A.: Grammar systems for distributed synthesis of approximate solutions extracted from experience. In: Paun, G., Salomaa, A. (eds.) Grammatical Models of Multi–Agent Systems, pp. 316–333. Gordon and Breach, Amsterdam (1999)
38. Polkowski, L., Skowron, A.: Towards an adaptive calculus of granules. In: Zadeh, L.A., Kacprzyk, J. (eds.) Computing with Words in Information/Intelligent Systems, vol. 1, pp. 201–228. Physica Verlag, Heidelberg (1999)
39. Proceedings of IEEE 2005 Conference on Granular Computing, GrC 2005, Beijing, China, July 2005. IEEE Press (2005)
40. Proceedings of IEEE 2006 Conference on Granular Computing, GrC 2006, Atlanta, USA, May 2006. IEEE Press, Los Alamitos (2006)
41. Liu, Q., Sun, H.: Theoretical study of granular computing. In: Wang, G.-Y., Peters, J.F., Skowron, A., Yao, Y. (eds.) RSKT 2006. LNCS (LNAI), vol. 4062, pp. 92–102. Springer, Heidelberg (2006)
42. Skowron, A., et al.: RSES: A system for data analysis, `http://logic.mimuw.edu.pl/~rses/`
43. Skowron, A., Rauszer, C.: The discernibility matrices and functions in decision systems. In: Słowiński, R. (ed.) Intelligent Decision Support. Handbook of Applications and Advances of the Rough Sets Theory, pp. 311–362. Kluwer, Dordrecht (1992)

44. Nguyen, S.H.: Regularity analysis and its applications in Data Mining. In: Polkowski, L., Tsumoto, S., Lin, T.Y. (eds.) Rough Set Methods and Applications, pp. 289–378. Physica Verlag, Heidelberg (2000)
45. Skowron, A.: Boolean reasoning for decision rules generation. In: Komorowski, J., Ras, Z. (eds.) ISMIS 1993. LNCS (LNAI), vol. 689, pp. 295–305. Springer, Heidelberg (1993)
46. Skowron, A.: Extracting laws from decision tables. Computational Intelligence. An International Journal 11(2), 371–388 (1995)
47. Skowron, A., Polkowski, L.: Synthesis of decision systems from data tables. In: Lin, T.Y., Cercone, N. (eds.) Rough Sets and Data Mining, pp. 289–299. Kluwer, Dordrecht (1997)
48. Skowron, A., Stepaniuk, J.: Information granules: towards foundations of granular computing. International Journal for Intelligent Systems 16, 57–85 (2001)
49. Stanfill, C., Waltz, D.: Toward memory–based reasoning. Communications of the ACM 29, 1213–1228 (1986)
50. Stanford Encyclopedia of Philosophy: Transworld Identity, `http://plato.stanford.edu/entries/~identity-transworld`
51. Stefanowski, J.: On rough set based approaches to induction of decision rules. In: Polkowski, L., Skowron, A. (eds.) Rough Sets in Knowledge Discovery, vol. 1, pp. 500–529. Physica Verlag, Heidelberg (1998)
52. Stepaniuk, J.: Knowledge discovery by application of rough set models. In: Polkowski, L., Tsumoto, S., Lin, T.Y. (eds.) Rough Set Methods and Applications, pp. 138–233. Physica Verlag, Heidelberg (2000)
53. `http://www.ics.uci.edu/~mlearn/databases/`
54. `http://www.ics.uci.edu.pl/~mlearn/MLSummary.html`
55. Wilcoxon, F.: Individual comparisons by ranking method. Biometrics 1, 80–83 (1945)
56. Wilson, D.R., Martinez, T.R.: Improved heterogeneous distance functions. Journal of Artificial Intelligence Research 6, 1–34 (1997)
57. Wojna, A.: Analogy–based reasoning in classifier construction. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets IV. LNCS, vol. 3700, pp. 277–374. Springer, Heidelberg (2005)
58. Wróblewski, J.: Covering with reducts – a fast algorithm for rule generation. In: Polkowski, L., Skowron, A. (eds.) RSCTC 1998. LNCS (LNAI), vol. 1424, pp. 402–407. Springer, Heidelberg (1998)
59. Wróblewski, J.: Adaptive aspects of combining approximation spaces. In: [18], pp. 139–156
60. Yao, Y.Y.: Information granulation and approximation in a decision–theoretic model of rough sets. In: [18], pp. 491–516
61. Yao, Y.Y.: Perspectives of granular computing. In: [39], pp. 85–90
62. Zadeh, L.A.: Fuzzy sets and information granularity. In: Gupta, M., Ragade, R., Yager, R.R. (eds.) Advances in Fuzzy Set Theory and Applications, pp. 3–18. North–Holland, Amsterdam (1979)
63. Zadeh, L.A.: Graduation and granulation are keys to computation with information described in natural language. In: [39], p. 30
64. Zeeman, E.C.: The topology of the brain and the visual perception. In: Fort, K.M. (ed.) Topology of 3–manifolds and Selected Topics, pp. 240–256. Prentice Hall, Upper Saddle River (1965)

On Classifying Mappings Induced by Granular Structures[⋆]

Lech Polkowski[1] and Piotr Artiemjew[2]

[1] Polish–Japanese Institute of Information Technology
Koszykowa 86, 02008 Warszawa, Poland
polkow@pjwstk.edu.pl
[2] University of Warmia and Mazury
Zolnierska 14, 10560 Olsztyn, Poland
artem@matman.uwm.edu.pl

To the memory of Professor Zdzisław Pawlak

Abstract. In this work the subject of granular computing is pursued beyond the content of the previous paper [21]. We study here voting on a decision by granules of training objects, granules of decision rules, granules of granular reflections of training data, and granules of decision rules induced from granular reflections of training data. This approach can be perceived as a direct mapping of the training data on test ones which is induced by granulation of knowledge on the training data. Some encouraging results were already presented in [21], and here the subject is pursued systematically.

Granules of knowledge are defined and computed according to a previously used scheme due to Polkowski in the framework of theory of rough inclusions.

On the basis of presented results, one is justified in concluding that the presented methods offer a very good quality of classification, comparable fully with best results obtained by other rough set based methods, like templates, adaptive methods, hybrid methods etc.

Keywords: rough inclusions, classification of data, granulation of knowledge, rough mereology.

1 Introduction: Rough Sets, Rough Inclusions, Granulation of Knowledge

Knowledge can be represented in the framework of rough set theory by means of information/decision systems, see [6], [7]. An *information system* is a pair $I = (U, A)$ where U is a set of objects, and A is a set of attributes; each attribute $a \in A$ a mapping on the set U into the value set V_a with $V = \bigcup_{a \in A} V_a$ as the global value set. A *decision system* is a pair $D = (I = (U, A)), d)$ where $d \notin A$

[⋆] This work is an extension of the work [20] presented by the authors at RSEISP07 International Conference.

is a *decision* with the value set V_d. Objects in U are represented by means of information sets: $Inf_A(u) = \{(a = a(u)) : a \in A\}$ is the information set of the object u; the formula $(a = a(u))$ is a particular case of a *descriptor* of the form $(a = v)$ where v is a value of the attribute $a \in A \cup \{d\}$. *Decision rules* are expressions of the form $\bigwedge_{a \in B}(a = a(u)) \Rightarrow (d = d(u))$ where $B \subseteq A$.

1.1 Indiscernibility, Granulation of Knowledge

Indiscernibility relations of the form $Ind(B) = \{(u, v) : a(u) = a(v)$ for each $a \in B\}$ where $B \subseteq A$, provide means for granulation of knowledge. The classical form of granulation of knowledge in information/decision systems is partitioning of U into classes of the indiscernibility relation $Ind(B)$. Each class $[u]_B = \{v \in U : (u, v) \in Ind(B)\}$ is interpreted as an elementary B–granule and unions of elementary B–granules are B–granules of knowledge. Thus, granulation means forming aggregates of objects indiscernible over chosen sets of attributes.

The approach to granulation presented in this work consists in using rough inclusions, see [12]–[19].

1.2 Rough Inclusions

The generic term of a *rough inclusion* was introduced in [22], see also [23]. A rough inclusion is a relation $\mu \subseteq U \times U \times [0, 1]$ which can be regarded as a graded similarity relation extending the indiscernibility relation by relaxing restrictions on attribute values, cf., [15].

In this work we are using prevalently rough inclusions obtained either from metrics or from continuous t–norms by means of their residual implications, see [17], [15], [14], [13], as well as their weak variants which in spite of violating some requirements for being a rough inclusion, have a clear intuitive appeal.

A rough inclusion μ, see, e.g., [23] or [12] and bibliography quoted there, can be defined as a relation on $U \times U \times [0, 1]$; the formula $\mu(u, v, r)$ reads "u is a part to degree $\geq r$ to v". A theoretical analysis and motivations for introducing a part relation into the realm of rough sets can be found, e.g., in [12], [15].

Granules of knowledge are defined, see [13], [14], [15], [16], from rough inclusions μ as follows (we depart here from mereological content, cf., e.g., [12], [14], [17], [19], giving a working definition of a granule),

$$g^{\mu}(v, r) \ \text{ is } \{u \in U : \mu(u, v, r)\}, \tag{1}$$

where $g^{\mu}(v, r)$ is the *granule of radius r about v* induced by the rough inclusion μ.

1.3 Rough Inclusions and Granules from Metrics

Given a bounded by 1 metric ρ on the set V of attribute values (we assume for convenience that one metric suits all attribute value sets), one lets, cf., [9] where the initial idea of a similarity was proposed,

$$\mu_{\rho}(u, v, r) \text{ iff } \rho(u, v) \leq 1 - r. \tag{2}$$

The relation μ_ρ has the properties that

Rh1. $\mu_\rho(u, v, 1)$ if and only if $u = v$.
Rh2. If $\mu_\rho(u, v, 1)$ then $\mu_\rho(w, u, r)$ implies $\mu_\rho(w, v, r)$ for each w.
Rh3. If $\mu_\rho(u, v, r)$ and $s < r$ then $\mu_\rho(u, v, s)$.

Properties Rh1–Rh3 are essentially requirements for a relation μ to be a rough inclusion.

In particular, see, e.g., [16], the Hamming metric H on information sets of objects in an information system, relative to size $|A|$ of the attribute set,

$$H(u, v) = \frac{|Inf_A(u) \triangle Inf_A(v)|}{|A|}, \tag{3}$$

where $X \triangle Y = (X \setminus Y) \cup (Y \setminus X)$ is the symmetric difference of sets X, Y, does induce the rough inclusion μ_H, according to (6), as,

$$\mu_H(u, v, r) \text{ iff } \frac{|Inf_A(u) \cap Inf_A(v)|}{|A|} \geq r, \tag{4}$$

meaning that u is a part to v (or, is similar to v) to degree not less than r if and only if at least $r \cdot 100$ percent of attributes in A, take the same value on u and v.

Granules induced by μ_H are of the form, according to (1),

$$g^H(v, r) \text{ is } \{u \in U : \frac{|Inf_A(u) \cap Inf_A(v)|}{|A|} \geq r\}, \tag{5}$$

i.e., an object u is in the granule $g_H(v, r)$ if and only if at least the fraction r of attributes agree on u and v.

1.4 A Graded Variant of μ_ρ

As defined in, e.g., [17], [18], the graded variant of μ_ρ assumes that in deciding the decision value at a test object u, collections of either rules or objects take part; these collections are built as granules with respect to a modified rough inclusion μ_ρ^ε, see [18].

Given $\varepsilon \in [0, 1]$, we let $\mu_\rho^\varepsilon(u, v, r)$ if and only if

$$|\{a \in A : \rho(a(v), a(u)) \leq \varepsilon\}| \geq r \cdot |A|. \tag{6}$$

With $\rho(x, y) = min\{1, |x - y|\}$, the granules induced by μ_ρ^ε are defined as,

$$g^{\mu_\rho^\varepsilon}(v, r) = \{u \in U : |\{a \in A : |a(v) - a(u)| \leq \varepsilon\}| \geq r \cdot |A|\}. \tag{7}$$

In particular, in case $r = 1$, the granule is defined as,

$$g^{\mu_\rho^\varepsilon}(v, 1) \text{ is } \{u \in U : |a(v) - a(u)| \leq \varepsilon \text{ for each} a \in A\}. \tag{8}$$

In computing granules, the parameter ε is normalized to the interval $[0, 1]$ as follows: first, for each attribute $a \in A$, the value $train(a) = max_{training\ set} a -$

$min_{training\ set}a$ is computed and the real line $(-\infty, +\infty)$ is contracted to the interval $[min_{training\ set}a, max_{training\ set}a]$ by the mapping f_a,

$$f_a(x) = \begin{cases} min_{training\ set}a \text{ in case } x \leq min_{training\ set}a \\ x \text{ in case } x \in [min_{training\ set}a, max_{training\ set}a] \\ max_{training\ set}a \text{ in case } x \geq max_{training\ set}a. \end{cases} \tag{9}$$

When the value $a(u)$ for a test object u is off the range

$$[min_{training\ set}a, max_{training\ set}a],$$

it is replaced with the value $f_a(a(u))$ in the range. The value of ε in this case is defined as $\frac{|f_a(a(u))-a(v)|}{train(a)}$.

Remark on duality between objects and decision rules. Objects in the universe of the information system $I = (U, A)$ are encoded by means of their information sets: an object u is represented by its information set $Inf_A(u) = \{(a = a(u)) : a \in A\}$. Decision rules in a decision system $D = (I = (U, A)), d)$ with the decision d, are written down as implications in the descriptor logic:

$$r : \bigwedge_{a \in A} (a = a(u)) \Rightarrow (d = d(u)).$$

Clearly, the mapping $K : Inf_A(u) \rightarrow \bigwedge_{a \in A}(a = a(u)) \Rightarrow (d = d(u))$ sends objects onto decision rules up to the indiscernibility and choice of decision values $d(u)$ in case of non–deterministic systems. Similarly, the mapping $L : \bigwedge_{a \in A}(a = a(u)) \Rightarrow (d = d(u)) \rightarrow Inf_A(u)$ sends decision rules onto objects uniquely up to indiscernibility.

Thus, formulas for granules of objects can be applied as well to decision rules: in effect, granules of decision rules can be formed whose centers are either decision rules or objects and vice versa, granules of objects can be formed about decision rules.

1.5 Granular Reflections of Granules and Granulated Data Sets

The idea of a granular reflection of a data set was proposed in [13]: for a chosen rough inclusion μ, given a granulation radius r, the set $G(r, \mu)$ of all granules of the radius r is formed. From this set, a covering $Cov(r, \mu, \mathcal{G})$ of the set of objects U is chosen by means of a strategy $\mathcal{G}$, which is usually a random choice of granules with irreducibility checking.

Given the covering $Cov(r, \mu, \mathcal{G})$, attributes in the set $A \cup \{d\}$ are factored through granules to make a new attribute set. For an attribute $a \in A \cup \{d\}$, and a granule g, the new attribute a^* is defined according to a chosen strategy $\mathcal{S}$ as

$$a^*(g) = \mathcal{S}(\{a(v) : v \in g\}). \tag{10}$$

A (possibly, but not necessarily) virtual object $o(g)$ is defined, called the *granular reflection* of g,

$$Inf_{A^*}o(g) = \{(a^* = a^*(g)) : a^* \in A^*, \tag{11}$$

where $A^* = \{a^* : a \in A\}$, which does represent the granule g.

A new information/decision system is formed:

$$(D^* = (I^* = (Cov(r, \mu, \mathcal{G}, A^*)), d^*))$$

called also a *granular reflection of the given information system* [13],[18].

In all tests reported here, the strategy $\mathcal{G}$ has been chosen as the random selection of a covering with checking for irreducibility and the strategy $\mathcal{S}$ has been the majority voting with random tie resolution.

2 Mapping Granules of Objects or Rules on Decision Values by Variants of μ_H

Given a granule g of either decision rules or objects in a decision system $D = (I = (U, A)), d)$, for each test object u, the value of decision assigned to u by the granule g is defined as,

$$\text{d(u)=c* iff } \frac{\text{sum of supports of rules pointing to c*}}{\text{cardinality of c* in the training set}} = max_c \frac{\text{sum of supports of rules pointing to c}}{\text{cardinality of c in the training set}}. \tag{12}$$

where c denotes a decision value and $c*$ is the decision value assigned to u. The last three rows of Table 1 show results obtained by our approach with granular systems, described in [16], [20], [21].

Table 1. Best results for Australian credit by some rough set based algorithms; in case $*$, reduction in object size is 49.9 percent, reduction in rule number is 54.6 percent; in case $**$, resp., 19.7, 18.2; in case $***$, resp., 3.6, 1.9

source	*method*	*accuracy*	*coverage*
[3]	SNAPM(0.9)	error=0.130	-
[5]	simple.templates	0.929	0.623
[5]	general.templates	0.886	0.905
[5]	closest.simple.templates	0.821	1.0
[5]	closest.gen.templates	0.855	1.0
[5]	tolerance.simple.templ.	0.842	1.0
[5]	tolerance.gen.templ.	0.875	1.0
[28]	adaptive.classifier	0.863	-
this.work	granular*.r=0.642857	0.8990	1.0
this.work	granular**.r=0.714826	0.964	1.0
this.work	granular***.concept.dependent.r=0.785714	0.9970	0.9995

2.1 Classifying Mappings Induced by $\mu_\rho^\varepsilon(u, v, 1)$

We present here results of tests in which the granulation with μ_ρ^ε with the radius $r = 1$ has been applied in four cases:

Case 1. Granules of training objects have been used.
Case 2. Granules of granular objects have been used.
Case 3. Granules of rules from the training set have been used.
Case 4. Granules of rules from the granulated training set have been used.

All tests have been done with Australian credit data [27], split into the training and test sets. The 5–fold cross validation has been applied. Results are expressed in terms of accuracy and coverage, cf., [26].

Table 1 shows the best results obtained with rough set based classification algorithms as well as augmenting them heuristics.

Case 1: Tests with granules of training objects. In Fig. 1 below the results of the test (test 1-v.1) are shown for accuracy and coverage respectively in function of ε applied. For comparison, the result obtained with the standard exhaustive classifier (marked with the horizontal line in Fig.1) is 0.845 for accuracy and 1.0 for coverage.

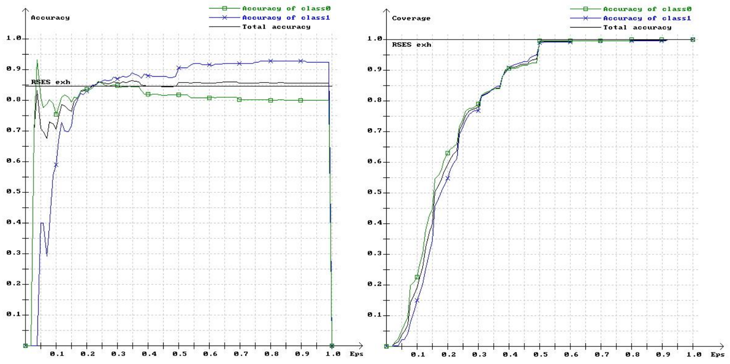


Fig. 1. Results for test with granules of training objects;Best result for $\varepsilon = 0.83$: accuracy = 0.859219; coverage = 0.998551

2.2 Case 2: Results of Tests with Granules of Granular Objects

In this case the parameters are ε and the granulation radius r. Results are shown in Table 2 where for each value of r, the optimal (best) result for ε's is given as *optimal eps.*

2.3 Case 3: Results of Tests with Granules of Rules from the Training Set

Fig. 2 shows the results.

Table 2. CV-5; Australian credit; Algorithm 2_v1. r_gran=granulation radius, optimal_eps=Best optimal epsilon, acc=Total accuracy, cov=Total coverage, m_trn=mean training sample. Best result for $r = 0.785714$ and $\varepsilon = 0.54$: accuracy=0.861673; coverage=0.9956.

r_gran	*optimal eps*	*acc*	*cov*	*m_trn*
nil	*nil*	0.845	1.0	552
0.000000	1.0	0.555073	1	1
0.071428	1.0	0.555073	1	1.2
0.142857	1.0	0.555073	1	2.2
0.214286	1.0	0.555073	1	3.4
0.285714	1.0	0.546377	1	5
0.357143	1.0	0.523189	1	9.6
0.428571	0.96	0.858144	0.742029	21.6
0.500000	0.95	0.838491	0.95942	52.8
0.571429	0.93	0.82871	0.998551	134.8
0.642857	0.95	0.831884	1.0	295.8
0.714286	0.71	0.858987	0.997102	456.4
0.785714	0.54	0.861673	0.995652	533.2
0.857143	0.83	0.859219	0.998551	546.2
0.928571	0.83	0.859219	0.998551	548
1.000000	0.83	0.859219	0.998551	552

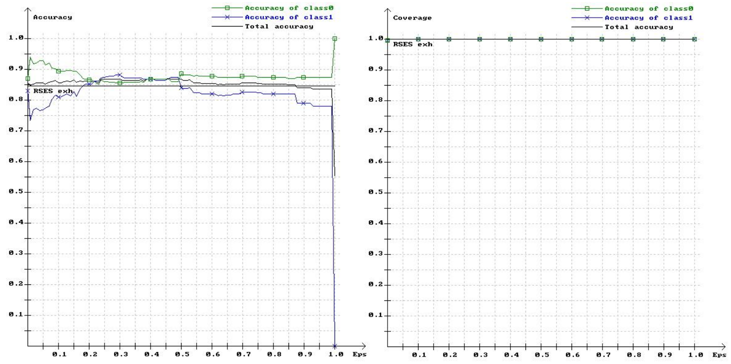


Fig. 2. Results for granules of rules from the training set. Best result for $\varepsilon = 0.46$: accuracy = 0.871015; coverage = 1.0.

2.4 Case 4: Results of Tests with Granules of Rules from the Granulated Training Set

Table 3 shows these results.

Conclusions. Results of presented tests show that classifying maps obtained from training objects give very satisfactory classification accuracy: the best result of accuracy of 0.871015, has been obtained in Case 2 with mapping granules of

Table 3. CV-5; Australian credit; Algorithm 4_v1. r_gran=granulation radius, optimal_eps=Best optimal epsilon, acc= accuracy, cov= coverage. Best result for $r = 0.785714$, $\varepsilon = 0.01$: accuracy=0.8507; coverage=1.0.

r_gran	*optimal eps*	*acc*	*cov*	*m_trn*
nil	*nil*	0.845	1.0	552
0	0	0.555073	1.0	1
0.071428	0	0.555073	1.0	1.6
0.142857	0	0.555073	1.0	2.2
0.214286	0.08	0.642029	1.0	2.8
0.285714	0	0.755073	1.0	4.4
0.357143	0	0.771014	1.0	9.8
0.428571	0.01	0.765217	1.0	23.8
0.500000	0	0.771014	1.0	53.2
0.571429	0	0.824638	1.0	130.6
0.642857	0	0.834783	1.0	294.8
0.714286	0.02	0.83913	1.0	454.8
0.785714	0.01	0.850724	1.0	533
0.857143	0.01	0.850724	1.0	546.2
0.928571	0.01	0.849275	1.0	548
1.000000	0.1	0.850725	1.0	552

rules from the training set; it is higher than almost all rough set method obtained results save for general and simple templates, granular test=train method and concept dependent granulation, see Table 1. In all cases, the best accuracy has been higher than that obtained with the standard exhaustive classifier.

These results support the intuitions [13] about effectiveness of granular structures in building classifiers.

2.5 Classifying Mappings Based on Granulation by Means of $\mu_\rho^\varepsilon(v, u, r)$

In this section, we present results of tests with granules based on $\mu_\rho^\varepsilon(v, u, r)$ for all values of r.We recall that in this case, a granule $g_{\mu_\rho^\varepsilon}(v, r)$ consists of objects uv such that at least $r \cdot 100$ percent of attributes a satisfy the condition $|a(u) - a(v)| \leq \varepsilon$. The parameter r is called the *catch radius.* We present results of experiments in three cases.

Case 1. Granules of training objects have been used.

Case 2. Granules of granular objects have been used.

Case 3. Granules of rules from the training set have been used.

Case 1: Results of tests with granules of training objects. Results are shown in Table 4.

Table 4. CV-5; Australian credit; Algorithm 1_v2. r_catch=catch radius, optimal_eps=Best optimal epsilon, acc=accuracy, cov= coverage. Best result for $r = 0.785714$, $\varepsilon = 0.18$: accuracy=0.872136; coverage=0.9971.

r_catch	*optimal eps*	*acc*	*cov*
nil	*nil*	0.845	1.0
0.071428	0	0.155073	1.0
0.142857	0	0.750725	1.0
0.214286	0.01	0.823188	1.0
0.285714	0.01	0.853623	1.0
0.357143	0.02	0.844927	1.0
0.428571	0.04	0.842029	1.0
0.500000	0.05	0.852174	1.0
0.571429	0.03	0.860869	1.0
0.642857	0.05	0.870866	0.998551
0.714286	0.24	0.868116	1.0
0.785714	0.18	0.872136	0.997102
0.857143	0.5	0.869565	1.0
0.928571	0.57	0.868116	1.0
1.000000	0.84	0.859219	0.998551

2.6 Case 2: Results of Tests with Granules of Granular Objects

These results are shown in Table 5.

Table 5. CV-5; Australian credit; Algorithm 2_v2. r_gran=granulation radius, optimal_r_catch=optimal catch radius, optimal_eps=Best optimal epsilon, acc=accuracy, cov=coverage. Best result for $r = 0.714826$, the optimal catch radius=0.714286, optimal ε=0.08: accuracy=0.874757; coverage=0.942.

r_gran	*optimal_r_catch*	*optimal eps*	*acc*	*cov*
0	0.357143	0.11	0.556755	0.997102
0.0714286	0.357143	0.11	0.556755	0.997102
0.142857	0.428571	0.15	0.556755	0.997102
0.214286	0.428571	0.11	0.570352	0.988406
0.285714	0.928571	0.91	0.739727	0.857971
0.357143	0.928571	0.92	0.790782	0.975362
0.428571	0.928571	0.87	0.797704	0.995652
0.5	0.785714	0.29	0.840527	0.989855
0.571429	0.642857	0.07	0.844695	0.998551
0.642857	0.642857	0.05	0.866476	0.998551
0.714286	0.714286	0.08	0.874757	0.994203
0.785714	0.785714	0.19	0.869417	0.998551
0.857143	0.785714	0.58	0.872464	1
0.928571	0.642857	0.05	0.872316	0.998551
1	0.785714	0.18	0.872136	0.997102

2.7 Case 3: Results of Tests with Granules of Rules from the Training Set

Table 6 shows results.

Table 6. CV-5; Australian credit; Algorithm 3_v2. r_catch=catch radius, optimal_eps=Best optimal epsilon, acc=accuracy, cov= coverage. Best result for r = 0.142857, ε=0.35: accuracy=0.868116; coverage=1.0.

r_catch	*optimal eps*	*acc*	*cov*
nil	*nil*	0.845	1.0
0	0	0.555073	1.0
0.071428	0	0.83913	1.0
0.142857	0.35	0.868116	1.0
0.214286	0.5	0.863768	1.0
0.285714	0.52	0.831884	1.0
0.357143	0.93	0.801449	1.0
0.428571	1.0	0.514493	1.0
0.500000	1.0	0.465217	1.0
0.571429	1.0	0.115942	1.0

Conclusions. In the cases with the catch radius r, accuracy is again higher than that of the standard exhaustive classifier. Best result is obtained in Case 2 with mappings of granules of granular objects: accuracy=0.87475 is higher than that obtained with closest templates, SNAPM, and adaptive classifier as recalled in Table 1.

3 Granulation by Means of Variants of Rough Inclusions Induced by Residual Implications of t–Norms

In this part of our work, we are using rough inclusions proposed in [13], see [14], [15], [18] as obtained from continuous t–norms by means of their residual implications.

For a continuous t–norm T, see, e.g., [4], [10], the residual implication $x \Rightarrow_T y$ is a mapping from the square $[0,1]^2$ into $[0,1]$ defined as follows,

$$x \Rightarrow_T y \geq z \text{ if and only if } T(x,z) \leq y; \tag{13}$$

thus, $x \Rightarrow_T y = max\{z : T(x,z) \leq y\}$.

As shown in, e.g., [13], [14], [15], [18], $\Rightarrow_T$ does induce a rough inclusion on the interval $[0,1]$:

$$\mu_{\rightarrow_T}(u,v,r) \text{ if and only if } x \Rightarrow_T y \geq r. \tag{14}$$

This rough inclusion can be transferred to the universe U of an information system as shown in [18]: for given objects u, v, and $\varepsilon \in [0,1]$, factors: $dis_\varepsilon(u,v) = \frac{|\{a \in A: |a(u)-a(v)| \geq \varepsilon\}|}{|A|}$, and $ind_\varepsilon(u,v) = \frac{|\{a \in A: |a(u)-a(v)| < \varepsilon\}|}{|A|}$ are introduced.

The weak variant of rough inclusion $\mu_{\rightarrow_T}$ is defined, see [18], as,

$$\mu^*_T(u,v,r) \text{ if and only if } dis_\varepsilon(u,v) \rightarrow_T \ ind_\varepsilon(u,v) \geq r. \tag{15}$$

Particular cases of this similarity measure induced by, respectively, t–norm $min(x,y)$, t–norm $P(x,y) = x \cdot y$, and t–norm $L(x,y) = max\{0, x+y-1\}$ are, see [18], [4],

For $T = min(x,y)$, $x \Rightarrow_{min} y$ is y in case $x > y$ and 1 otherwise hence $\mu^*_{min}(u,v,r)$ if and only if $dis_\varepsilon(u,v) > ind_\varepsilon(u,v) \geq r$ with $r < 1$ and 1 otherwise.

For $t = P$ where $P(x,y) = x \cdot y$, $x \Rightarrow_P y = \frac{y}{x}$ when $x \neq 0$ and 1 when $x = 0$ hence $\mu^*_P(u,v,r)$ if and only if $\frac{ind_\varepsilon(u,v)}{dis_\varepsilon(u,v)} \geq r$ with $r < 1$ and 1 otherwise.

For $t = L$, $x \Rightarrow_L y = min\{1, 1-x+y\}$, hence $\mu^*_L(u,v,r)$ if and only if $1 - dis_\varepsilon(u,v) + ind_\varepsilon(u,v) \geq r$ with $r < 1$ and 1 otherwise.

These similarity measures will be applied in building granules and then in data classification.

Tests have been carried out with Australian credit data set [27] and the method was CV-5 (the 5–fold cross validation).

3.1 Classifying Mappings Based on Granulation by Means of $\mu^*_T(u,v,r)$

In this Section results of tests are presented with granulation based on the weak rough inclusion $\mu^*_T(u,v,r)$. For each of t–norms: min, P, L, we have three cases:

Case 1. Granules of training objects have been used.

Case 2. Granules of rules induced from the training set have been used.

Case 3. Granules of granular objects induced from the training set have been used.

3.2 Case 1: Results of Tests with Granules of Training Objects

In this approach, training objects are made into granules for a given ε. Objects in each granule g about a test object u, vote for decision value at u as follows: for each decision class c, the value $\text{p(c)} = \frac{\sum \text{training object v in g falling in c}^{\,w(v,t)}}{\text{size of c in training set}}$ is computed where the weight $w(v,t)$ is computed for a given t–norm T as $w(v,t) = dis_\varepsilon(u,v) \rightarrow_T \ ind_\varepsilon(u,v)$. The class c* assigned to u is the one with the largest value of p. Results for the three chosen t–norms are given in Fig.1 (t=min), Fig.2 (t=P), Fig.3 (t=L). For comparison, the accuracy computed with the standard exhaustive classifier is 0.845, and coverage is 1.0.

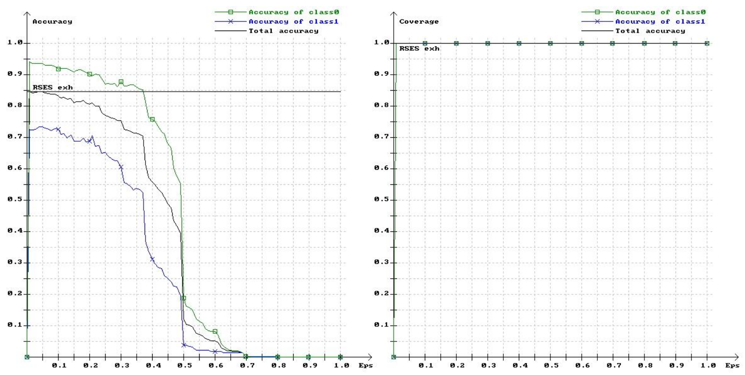


Fig. 3. Results for granules of training objects,T=min.Best result for $\varepsilon = 0.04$: accuracy = 0.847826, coverage = 1.0

Conclusions. For all three t–norms, accuracy in the best case is better than that obtained with the standard exhaustive classifier, though is is lower than accuracy obtained with the rough inclusion μ_H. One may conclude that the impact of weighting by means of μ_T^* is slightly less decisive than the impact of direct comparing of values of attributes on objects by means of the threshold parameter ε.

3.3 Case 2: Results of Tests with Granules of Rules from the Training Set

Weighted voting of rules in a given granule g for decision at test object u goes according to the formula d(u)= arg max p(c) where

$$\text{p(c)}=\frac{\Sigma\text{rule in g pointing to c}^{\,w(r,t)\cdot support(r)}}{\text{size of c in training set}},$$

where weight $w(r,t)$ is computed as $dis_\varepsilon(u,r) \rightarrow_T \ ind_\varepsilon(u,r)$.

Results are shown in Fig. 4 (T=min), Fig. 5 (T=P), Fig.6 (T=L).

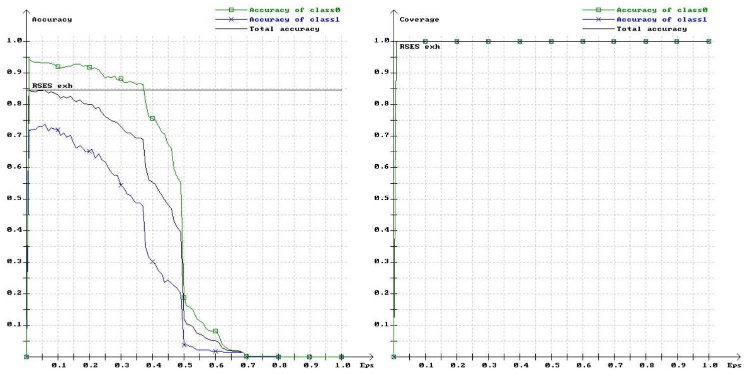


Fig. 4. Results for granules of training objects,T=P.Best result for $\varepsilon = 0.06$: accuracy = 0.847826, coverage = 1.0

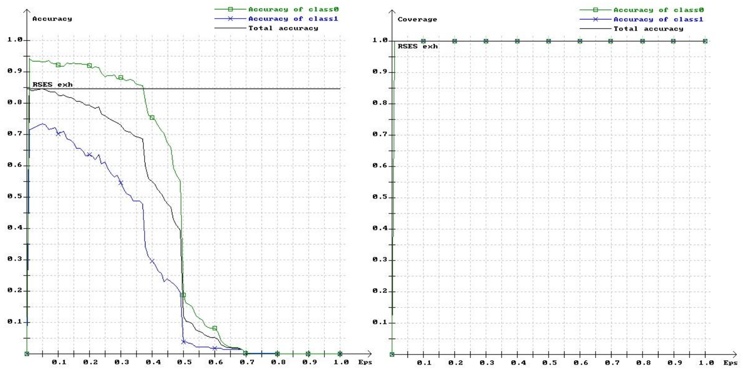


Fig. 5. Results for granules of training objects, T=L. Best result for $\varepsilon = 0.05$: accuracy $= 0.846377$, coverage $= 1.0$.

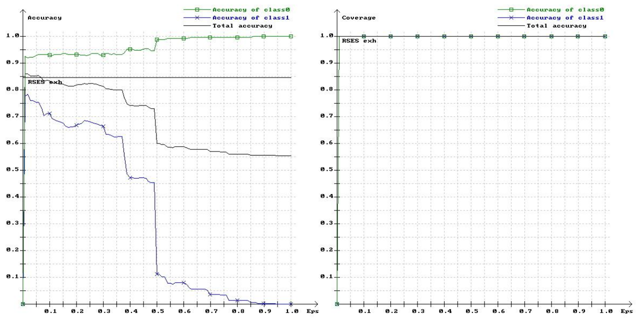


Fig. 6. Results for granules of rules, T=min. Best result for $\varepsilon = 0.02$: accuracy $=$ 0.86087, coverage $= 1.0$.

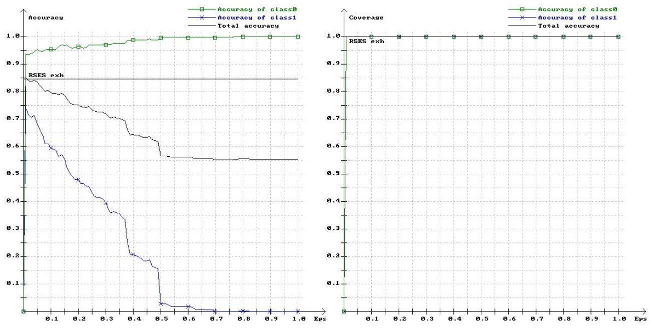


Fig. 7. Results for granules of rules,T=P. Best result for $\varepsilon = 0.01$: accuracy $= 0.850725$, coverage $= 1.0$.

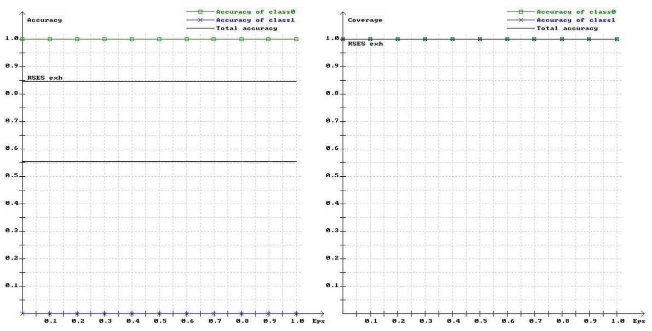


Fig. 8. Results for granules of rules, T=L. Best result for $\varepsilon = 0$, accuracy = 0.555073, coverage = 1.0.

Conclusions. For t–norms *min* and *P* classification accuracy is higher than that of the standard exhaustive classifier and case of *min* is better than all cases with granules of training objects. The case of *L* is worse which can be attributed to the computational features of $\Rightarrow_L$: weights in this case are little discriminating.

3.4 Results of Tests with Granular Objects from the Training Set

Analogously, as with granules of training objects, granular objects from granular reflections of granules vote for decision. The difference is in the fact that now we

Table 7. CV-5; Australian credit; Algorithm 5_v3. Granular objects, T=min; r_gran= granulation radius, optimal_eps= optimal epsilon, acc= accuracy, cov=coverage, m_trn=mean training set. Best result for $r = 0.785714$, *varepsilon*=0.05: accuracy=0.855072; coverage=1.0.

r_gran	*optimal eps*	*acc*	*cov*	*m_trn*
nil	*nil*	0.845	1.0	552
0	0.01	0.555073	1.0	1
0.071428	0.01	0.555073	1.0	1.4
0.142857	0.01	0.555073	1.0	2
0.214286	0.01	0.555073	1.0	2.4
0.285714	0.01	0.531884	1.0	4.8
0.357143	0.01	0.743478	1.0	9.2
0.428571	0.01	0.733333	1.0	20.4
0.500000	0.03	0.834783	1.0	53.8
0.571429	0.02	0.791304	1.0	134.4
0.642857	0.01	0.798551	1.0	295.8
0.714286	0.02	0.83913	1.0	454.8
0.785714	0.05	0.855072	1.0	533.8
0.857143	0.05	0.847826	1.0	546.2
0.928571	0.04	0.847826	1.0	548
1.000000	0.04	0.847826	1.0	552

Table 8. CV-5; Australian credit; Algorithm 6_v3. Granular objects, T=P; r_gran=granulation radius, optimal_eps= optimal epsilon, acc= accuracy, cov= coverage, m_trn=mean training set. Best results for $r = 0.785714$, ε=0.01: accuracy=0.852174; coverage=1.0.

r_gran	*optimal eps*	*acc*	*cov*	*m_trn*
nil	*nil*	0.845	1.0	552
0	0.01	0.555073	1.0	1
0.071428	0.01	0.555073	1.0	1.2
0.142857	0.01	0.555073	1.0	2.2
0.214286	0.01	0.562319	1.0	2.8
0.285714	0.01	0.585507	1.0	4.8
0.357143	0.01	0.594203	1.0	10.2
0.428571	0.01	0.728986	1.0	22.4
0.500000	0.01	0.808696	1.0	54.8
0.571429	0.01	0.746377	1.0	131.8
0.642857	0.01	0.763768	1.0	295.2
0.714286	0.01	0.818841	1.0	454.4
0.785714	0.01	0.852174	1.0	533.2
0.857143	0.01	0.847826	1.0	546.2
0.928571	0.01	0.846377	1.0	548
1.000000	0.06	0.847826	1.0	552

Table 9. CV-5; Australian credit; Algorithm 7_v3. Granular objects, T=L; r_gran=granulation radius, optimal_eps= optimal epsilon, acc=accuracy, cov=coverage, m_trn=mean training set. Best result for $r = 0.785714$, ε=0.01: accuracy=0.85942; coverage=1.0.

r_gran	*optimal eps*	*acc*	*cov*	*m_trn*
nil	*nil*	0.845	1.0	552
0	0.01	0.555073	1.0	1
0.071428	0.01	0.555073	1.0	1.2
0.142857	0.01	0.555073	1.0	1.4
0.214286	0.01	0.555073	1.0	3.4
0.285714	0.01	0.513043	1.0	4.6
0.357143	0.01	0.511594	1.0	8.8
0.428571	0.01	0.666667	1.0	22.8
0.500000	0.01	0.707247	1.0	53.2
0.571429	0.01	0.595652	1.0	132
0.642857	0.01	0.563768	1.0	292.2
0.714286	0.02	0.786956	1.0	457.6
0.785714	0.01	0.85942	1.0	533
0.857143	0.05	0.847826	1.0	546.2
0.928571	0.05	0.849275	1.0	548
1.000000	0.05	0.846377	1.0	552

have two–parameter case with ε, r hence results are given in Table 7 (T=min), Table 8 (T=P), Table 9 (T=L) in which for each row corresponding to the radius of granulation, the best result for ε is given along with accuracy and coverage in that case as well as the value *optimal eps*.

4 Weighted Voting by Granules of Training Objects According to $\mu_\rho^\varepsilon(u, v, 1)$

The rough inclusion $\mu_\rho^\varepsilon(u, v, r)$ has been introduced in (6); in particular, in (8), the variant $\mu_\rho^\varepsilon(u, v, 1)$ has been specialized.

In this part, we proceed with weighted voting; the difference in comparison with the approach in sect. 3.1 consists in modifying the weight computing scheme, in order to increase weights in case the difference in attribute values does exceed the threshold of ε and to decrease weights in the contrary case. This idea is here implemented in five variants (as Algorithms 8_v1.1, v1.2, v1.3, v1.4, v1.5). As with the former experiments, $max_{training\ set}a$ and $min_{training\ set}a$ have been found from original training data set for each attribute a and outliers in the test set have been projected onto $max_{training\ set}a$ and $min_{training\ set}a$ for each attribute a (see sect. 1.4).

Classification of testing objects by means of weighted granules of training objects has been done as follows:

1. A chosen value of ε (determining attribute similarity) has been input.

For all attributes a_k, where $k = 1..number\ of\ conditional\ attributes$; we compute:

For m=1..number of training objects, n=1..number of testing objects;
Case 1. (Algorithm 8_v1.1)

$w(u_n, v_m) = w(u_n, v_m) + \frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k|}$.

Case 2. (Algorithm 8_v1.2)

1) If $\frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k|} \geq \varepsilon \mapsto$

$w(u_n, v_m) = w(u_n, v_m) + \frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k|} * \frac{(1+\varepsilon)}{\varepsilon}$;

2) If $\frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k|} < \varepsilon \mapsto$

$w(u_n, v_m) = w(u_n, v_m) + \frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k|} * \frac{1}{\varepsilon}$.

Case 3. (Algorithm 8_v1.3)

1) If $\frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k|} \geq \varepsilon \mapsto$

$w(u_n, v_m) = w(u_n, v_m) + \frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k|} * (1 + \varepsilon)$;

2) If $\frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k|} < \varepsilon \mapsto$

$w(u_n,v_m) = w(u_n,v_m) + \frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k|} * \frac{1}{1+\varepsilon}$.

Case 4. (Algorithm 8_v1.4)

1) If $\frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k|} \geq \varepsilon \mapsto$
$w(u_n,v_m) = w(u_n,v_m) + \frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k| * (\varepsilon + \frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k|})}$, ie.

$w(u_n,v_m) = w(u_n,v_m) + \frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k| * \varepsilon + |a_k(u_n)-a_k(v_m)|}$;

2) If $\frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k|} < \varepsilon \mapsto w(u_n,v_m) = w(u_n,v_m) + \frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k| * \varepsilon}$.

Case 5. (Algorithm 8_v1.5.)

1) If $\frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k|} \geq \varepsilon \mapsto w(u_n,v_m) = w(u_n,v_m) + \frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k|}$;

2) If $\frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k|} < \varepsilon \mapsto w(u_n,v_m) = w(u_n,v_m) + \frac{|a_k(u_n)-a_k(v_m)|}{|max\_attr_k - min\_attr_k| * \varepsilon}$.

After computing weights in either case is completed for a given test object u_n and each training object v_m, the voting procedure consists in computing values of parameters,

$$Param1 = \frac{\sum_{v_m \text{ in positive class}} w(u_n,v_m)}{\text{cardinality of positive class}}.$$

$$Param2 = \frac{\sum_{v_m \text{ in negative class}} w(u_n,v_m)}{\text{cardinality of negative class}}.$$

If $Param1 < Param2$ then the test object u_n is classified to the positive class, otherwise it is classified into the negative class. After all test objects u_n are classified, quality parameters accuracy and coverage are computed,

$$acc = \frac{\text{number of correctly classified objects}}{\text{number of classified objects}},$$

$$cov = \frac{\text{number of classified objects}}{\text{number of test objects}}.$$

Results of experiments with these variants of classifiers are given in Tables 10, 11,12,13,14 for Australian credit data set already tested with former experiments. The method applied was CV-5 (5–fold cross validation).

For comparison, we performed CV–5 tests with this data set and the standard exhaustive classifier as well as with k–nn method as implemented in the RSES system [26], shown in Table 10. The results for k–nn method shown in Table 10 are best over all possible parameter values. The best result for RSES implemented $k - nn$ method was $accuracy = 0.859$ and for the exhaustive classifier,

Table 10. CV–5 for Australian credit data set (15attr, 690obj) for Algorithm 8_v1.1

result for	$FOLD1$	$FOLD2$	$FOLD3$	$FOLD4$	$FOLD5$	$CV-5$
	T_acc	*T_acc*	*T_acc*	*T_acc*	*T_acc*	*T_acc*
RSES exh	0.848	0.848	0.848	0.862	0.819	0.845
RSES k-nn	0.862	0.855	0.884	0.841	0.855	0.859
Alg 8_v 1.1	0.884058	0.876812	0.862319	0.862319	0.862319	0.869565

Table 11. CV–5 for Australian credit data set (15attr, 690obj) for Algorithm 8_v1.2

epsilon	$FOLD1$	$FOLD2$	$FOLD3$	$FOLD4$	$FOLD5$	$CV-5$
	T_acc	*T_acc*	*T_acc*	*T_acc*	*T_acc*	*T_acc*
RSES exh	0,848	0,848	0,848	0,862	0,819	0,845
$RSES\ k-nn$	0,862	0,855	0,884	0,841	0,855	0,859
0,01..0,1	0,884058	0,876812	0,862319	0,862319	0,862319	0,869565
0,11..0,15	0,884058	0,876812	0,862319	0,869565	0,862319	0,871015
0,16..0,25	0,876812	0,884058	0,862319	0,862319	0,862319	0,869565
0,26	0,876812	0,876812	0,862319	0,862319	0,862319	0,868116
0,27	0,876812	0,869565	0,862319	0,862319	0,862319	0,866667
0,28..0,29	0,876812	0,876812	0,862319	0,862319	0,862319	0,868116
0,3	0,876812	0,884058	0,862319	0,862319	0,862319	0,869565
0,31	0,869565	0,876812	0,862319	0,862319	0,862319	0,866667
0,32	0,869565	0,869565	0,855072	0,862319	0,862319	0,863768
0,33..0,37	0,869565	0,876812	0,855072	0,862319	0,862319	0,865217
0,38..0.39	0,869565	0,876812	0,855072	0,862319	0,869565	0,866667
0,4..0,46	0,869565	0,884058	0,855072	0,862319	0,869565	0,868116
0,47..0,49	0,876812	0,884058	0,855072	0,862319	0,869565	0,869565
0,5..0,59	0,869565	0,869565	0,855072	0,862319	0,862319	0,863768
0,6	0,869565	0,869565	0,855072	0,869565	0,862319	0,865217
0,61	0,876812	0,869565	0,855072	0,869565	0,862319	0,866667
0,62	0,876812	0,869565	0,862319	0,869565	0,869565	0,869565
0,63	0,876812	0,869565	0,869565	0,869565	0,869565	0,871014
0,64..0,67	0,869565	0,869565	0,869565	0,869565	0,869565	0,869565
0,68..0,69	0,862319	0,862319	0,869565	0,869565	0,869565	0,866667
0,7	0,876812	0,862319	0,869565	0,869565	0,869565	0,869565
0,71..0,72	0,884058	0,862319	0,869565	0,869565	0,869565	0,871014
0,73..0,76	0,884058	0,855072	0,869565	0,869565	0,869565	0,869565
0,77..0,84	0,884058	0,855072	0,869565	0,862319	0,869565	0,868116
0,85..0,89	0,876812	0,855072	0,869565	0,862319	0,869565	0,866667
0,9..0,91	0,869565	0,847826	0,876812	0,862319	0,869565	0,865217
0,92..0,99	0,876812	0,847826	0,876812	0,862319	0,869565	0,866667
1	0,884058	0,876812	0,862319	0,862319	0,862319	0,869565

$accuracy = 0.845$ (coverage in either case was found to be 1.0). Optimal results given below for variants 1 to 5 of our approach are always better.

Table 10 shows results of CV–5 test in Case 1 (Algorithm 8_v1.1): we obtain $accuracy = 0.869565$, $coverage = 1.0$.

For variants 2–5, we introduce ε for 100 values from 0 to 1. For each of these values, weights have been computed as described above and voting has taken place.

Table 11 shows results in Case 2: (Algorithm 8_v1.2; best result is obtained for ε in the range 0.11 – 0.15: $accuracy = 0,871015$, $coverage = 1.0$.

Table 12 shows results in Case 3 (Algorithm 8_v1.3); best result is obtained for ε in the range 0.1 – 0.13: $accuracy = 0,872464$, $coverage = 1.0$.

Table 12. CV–5 for Australian credit data set (15attr, 690obj) for Algorithm 8_v1.3

epsilon	$FOLD1$ T_acc	$FOLD2$ T_acc	$FOLD3$ T_acc	$FOLD4$ T_acc	$FOLD5$ T_acc	$CV-5$ T_acc
$RSES\ exh$	0,848	0,848	0,848	0,862	0,819	0,845
$RSES\ k-nn$	0,862	0,855	0,884	0,841	0,855	0,859
0,01 − 0,08	0,884058	0,876812	0,862319	0,862319	0,862319	0,869565
0,09	0,884058	0,884058	0,862319	0,862319	0,862319	0,871015
0,1 − 0,13	0,884058	0,884058	0,862319	0,869565	0,862319	0,872464
0,14 − 0,15	0,884058	0,884058	0,862319	0,862319	0,862319	0,871015
0,16 − 0,21	0,876812	0,884058	0,862319	0,862319	0,862319	0,869565
0,22	0,876812	0,876812	0,862319	0,862319	0,862319	0,868116
0,23 − 0,24	0,876812	0,869565	0,862319	0,862319	0,862319	0,866667
0,25	0,876812	0,862319	0,862319	0,862319	0,862319	0,865218
0,26	0,876812	0,869565	0,862319	0,862319	0,862319	0,866667
0,27	0,876812	0,869565	0,855072	0,862319	0,862319	0,865217
0,28	0,869565	0,869565	0,855072	0,862319	0,862319	0,863768
0,29 − 0,35	0,869565	0,876812	0,855072	0,862319	0,862319	0,865217
0,35 − 0,37	0,869565	0,876812	0,855072	0,862319	0,869565	0,866667
0,38	0,869565	0,876812	0,855072	0,855072	0,869565	0,865217
0,39 − 0,46	0,876812	0,876812	0,855072	0,855072	0,869565	0,866667
0,47 − 0,49	0,869565	0,876812	0,855072	0,855072	0,869565	0,865217
0,5 − 0,52	0,884058	0,869565	0,862319	0,862319	0,862319	0,868116
0,53	0,884058	0,869565	0,862319	0,862319	0,869565	0,869565
0,54	0,891304	0,862319	0,869565	0,869565	0,869565	0,872464
0,55 − 0,61	0,891304	0,855072	0,869565	0,869565	0,869565	0,871014
0,62 − 0,63	0,884058	0,855072	0,869565	0,855072	0,869565	0,866666
0,64 − 0,65	0,876812	0,855072	0,869565	0,855072	0,869565	0,865217
0,66 − 0,68	0,869565	0,855072	0,869565	0,855072	0,869565	0,863768
0,69	0,869565	0,847826	0,869565	0,855072	0,869565	0,862319
0,7 − 0,71	0,862319	0,847826	0,869565	0,855072	0,869565	0,860869
0,72 − 0,74	0,862319	0,855072	0,869565	0,855072	0,869565	0,862319
0,75	0,862319	0,847826	0,869565	0,855072	0,869565	0,860869
0,76	0,862319	0,855072	0,869565	0,855072	0,869565	0,862319
0,77 − 0,78	0,847826	0,855072	0,876812	0,855072	0,869565	0,860869
0,79	0,847826	0,855072	0,884058	0,855072	0,869565	0,862319
0,8 − 0,84	0,847826	0,847826	0,884058	0,855072	0,869565	0,860869
0,85 − 0,87	0,847826	0,840580	0,884058	0,855072	0,869565	0,859420
0,88 − 0,92	0,840580	0,840580	0,884058	0,855072	0,869565	0,857971
0,93 − 0,99	0,840580	0,840580	0,884058	0,847826	0,869565	0,856522
1	0,884058	0,876812	0,862319	0,862319	0,862319	0,869565

Table 13 shows results in Case 4 (Algorithm 8_v1.4); best result is obtained for $\varepsilon = 1.0$: $accuracy = 0.869565$, $coverage = 1.0$.

Table 14 shows results in Case 5 (Algorithm 8_v1.5); best result is obtained for $\varepsilon = 0.13$: in the range $0.1 - 0.13$: $accuracy = 0.87971$, $coverage = 1.0$.

Table 13. CV–5 for Australian credit data set (15attr, 690obj) for Algorithm 8_v1.4

$epsilon$	$FOLD1$	$FOLD2$	$FOLD3$	$FOLD4$	$FOLD5$	$CV-5$
	T_acc	T_acc	T_acc	T_acc	T_acc	T_acc
$RSES\ exh$	0,848	0,848	0,848	0,862	0,819	0,845
$RSES\ k-nn$	0,862	0,855	0,884	0,841	0,855	0,859
0,01	0,862319	0,826087	0,862319	0,847826	0,855072	0,850725
0,02	0,855072	0,833333	0,862319	0,847826	0,847826	0,849275
0,03	0,862319	0,833333	0,862319	0,855072	0,855072	0,853623
0,04	0,855072	0,840580	0,869565	0,847826	0,847826	0,852174
0,05	0,862319	0,826087	0,869565	0,847826	0,855072	0,852174
0,06	0,862319	0,833333	0,869565	0,855072	0,855072	0,855072
0,07	0,862319	0,826087	0,869565	0,869565	0,862319	0,857971
0,08	0,862319	0,826087	0,862319	0,876812	0,855072	0,856522
0,09	0,862319	0,833333	0,862319	0,876812	0,855072	0,857971
0,1	0,862319	0,833333	0,862319	0,876812	0,855072	0,857971
0,11 − 0,13	0,869565	0,833333	0,869565	0,884058	0,855072	0,862319
0,14	0,869565	0,840580	0,869565	0,876812	0,855072	0,862319
0,15	0,869565	0,833333	0,876812	0,869565	0,855072	0,860869
0,16	0,876812	0,833333	0,876812	0,884058	0,847826	0,863768
0,17	0,869565	0,833333	0,876812	0,869565	0,855072	0,860869
0,18	0,869565	0,833333	0,862319	0,869565	0,855072	0,857971
0,19	0,869565	0,826087	0,869565	0,869565	0,855072	0,857971
0,2	0,869565	0,818841	0,869565	0,869565	0,855072	0,856522
0,21 − 0,22	0,869565	0,818841	0,869565	0,862319	0,855072	0,855072
0,23	0,869565	0,811594	0,869565	0,862319	0,855072	0,853623
0,24	0,869565	0,818841	0,862319	0,862319	0,855072	0,853623
0,25	0,855072	0,818841	0,862319	0,862319	0,855072	0,850725
0,26	0,862319	0,811594	0,855072	0,862319	0,855072	0,849275
0,27	0,847826	0,811594	0,855072	0,855072	0,855072	0,844927
0,28 − 0,29	0,862319	0,811594	0,855072	0,862319	0,855072	0,849275
0,3	0,855072	0,804348	0,862319	0,862319	0,855072	0,847826
0,31	0,847826	0,797101	0,855072	0,862319	0,847826	0,842029
0,32	0,847826	0,797101	0,862319	0,862319	0,847826	0,843478
0,33	0,847826	0,804348	0,869565	0,862319	0,840580	0,844928
0,34	0,847826	0,818841	0,862319	0,862319	0,833333	0,844928
0,35	0,862319	0,818841	0,862319	0,862319	0,833333	0,847826
0,36	0,862319	0,826087	0,862319	0,869565	0,833333	0,850725
0,37	0,855072	0,833333	0,855072	0,869565	0,840580	0,850724
0,38	0,862319	0,804348	0,869565	0,884058	0,840580	0,852174
0,39	0,855072	0,804348	0,862319	0,884058	0,840580	0,849275
0,4	0,855072	0,804348	0,876812	0,884058	0,840580	0,852174
0,41	0,855072	0,811594	0,869565	0,884058	0,840580	0,852174
0,42 − 0,43	0,855072	0,818841	0,869565	0,884058	0,840580	0,853623
0,44	0,855072	0,811594	0,869565	0,884058	0,833333	0,850724
0,45	0,855072	0,811594	0,876812	0,891304	0,833333	0,853623
0,46	0,855072	0,818841	0,876812	0,891304	0,833333	0,855072
0,47	0,855072	0,818841	0,869565	0,898551	0,833333	0,855072
0,48	0,862319	0,818841	0,869565	0,898551	0,833333	0,856522
0,49 − 0,52	0,862319	0,826087	0,869565	0,898551	0,833333	0,857971
0,53	0,855072	0,833333	0,869565	0,891304	0,833333	0,856521
0,54 − 0,6	0,862319	0,826087	0,869565	0,898551	0,826087	0,856522
0,61	0,862319	0,833333	0,869565	0,898551	0,826087	0,857971
0,62	0,855072	0,840580	0,876812	0,905797	0,840580	0,863768
0,63	0,855072	0,840580	0,876812	0,905797	0,833333	0,862319
0,64 − 0,65	0,855072	0,840580	0,876812	0,898551	0,833333	0,860870
0,66 − 0,67	0,855072	0,840580	0,876812	0,898551	0,840580	0,862319
0,68 − 0,76	0,855072	0,840580	0,876812	0,898551	0,847826	0,863768
0,77 − 0,8	0,862319	0,833333	0,876812	0,905797	0,847826	0,865217
0,81 − 0,82	0,862319	0,833333	0,876812	0,905797	0,855072	0,866667
0,83 − 0,84	0,862319	0,833333	0,876812	0,898551	0,855072	0,865217
0,85	0,862319	0,833333	0,884058	0,898551	0,855072	0,866667
0,86 − 0,87	0,862319	0,833333	0,876812	0,898551	0,855072	0,865217
0,88 − 0,91	0,862319	0,840580	0,869565	0,891304	0,855072	0,863768
0,92 − 0,93	0,855072	0,840580	0,869565	0,891304	0,855072	0,862319
0,94 − 0,96	0,855072	0,847826	0,869565	0,891304	0,855072	0,863768
0,97 − 0,99	0,855072	0,847826	0,876812	0,891304	0,855072	0,865217
1	0,884058	0,876812	0,862319	0,862319	0,862319	0,869565

Table 14. CV–5 for Australian credit data set (15attr, 690obj) for Algorithm 8_*v*1.5

epsilon	*FOLD*1 *T_acc*	*FOLD*2 *T_acc*	*FOLD*3 *T_acc*	*FOLD*4 *T_acc*	*FOLD*5 *T_acc*	*CV* − 5 *T_acc*
RSES exh	0,848	0,848	0,848	0,862	0,819	0,845
RSES k − *nn*	0,862	0,855	0,884	0,841	0,855	0,859
0,01	0,884058	0,876812	0,847826	0,855072	0,855072	0,863768
0,02	0,876812	0,847826	0,840580	0,862319	0,855072	0,856522
0,03	0,869565	0,826087	0,811594	0,847826	0,869565	0,844927
0,04	0,855072	0,840580	0,818841	0,855072	0,869565	0,847826
0,05	0,826087	0,847826	0,840580	0,869565	0,869565	0,850725
0,06	0,855072	0,869565	0,855072	0,876812	0,862319	0,863768
0,07	0,862319	0,869565	0,855072	0,876812	0,869565	0,866667
0,08	0,862319	0,869565	0,876812	0,862319	0,876812	0,869565
0,09	0,855072	0,869565	0,876812	0,855072	0,862319	0,863768
0,1	0,862319	0,884058	0,891304	0,869565	0,869565	0,875362
0,11	0,862319	0,884058	0,891304	0,862319	0,884058	0,876812
0,12	0,869565	0,876812	0,884058	0,862319	0,876812	0,873913
0,13	0,869565	0,876812	0,884058	0,876812	0,891304	0,879710
0,14	0,869565	0,862319	0,876812	0,869565	0,898551	0,875362
0,15	0,869565	0,855072	0,884058	0,869565	0,898551	0,875362
0,16	0,884058	0,855072	0,869565	0,869565	0,876812	0,871014
0,17	0,884058	0,855072	0,876812	0,876812	0,876812	0,873913
0,18	0,884058	0,847826	0,847826	0,884058	0,876812	0,868116
0,19	0,876812	0,847826	0,869565	0,876812	0,869565	0,868116
0,2	0,869565	0,847826	0,869565	0,891304	0,869565	0,869565
0,21	0,876812	0,855072	0,862319	0,891304	0,862319	0,869565
0,22	0,876812	0,855072	0,869565	0,884058	0,862319	0,869565
0,23	0,876812	0,847826	0,869565	0,876812	0,869565	0,868116
0,24	0,869565	0,840580	0,884058	0,869565	0,862319	0,865217
0,25	0,869565	0,847826	0,869565	0,862319	0,869565	0,863768
0,26	0,869565	0,847826	0,869565	0,869565	0,869565	0,865217
0,27	0,869565	0,840580	0,869565	0,869565	0,869565	0,863768
0,28	0,869565	0,840580	0,884058	0,869565	0,855072	0,863768
0,29	0,869565	0,840580	0,891304	0,869565	0,855072	0,865217
0,3	0,869565	0,818841	0,869565	0,876812	0,855072	0,857971
0,31	0,862319	0,833333	0,876812	0,869565	0,847826	0,857971
0,32	0,869565	0,833333	0,869565	0,884058	0,847826	0,860869
0,33	0,862319	0,833333	0,876812	0,876812	0,840580	0,857971
0,34	0,862319	0,840580	0,876812	0,884058	0,840580	0,860870
0,35 − 0,37	0,862319	0,826087	0,876812	0,905797	0,840580	0,862319
0,38	0,869565	0,811594	0,876812	0,884058	0,847826	0,857971
0,39	0,862319	0,826087	0,876812	0,891304	0,855072	0,862319
0,4 − 0,42	0,862319	0,826087	0,876812	0,891304	0,847826	0,860870
0,43	0,862319	0,826087	0,884058	0,891304	0,847826	0,862319
0,44	0,862319	0,818841	0,884058	0,891304	0,847826	0,860870
0,45 − 0,46	0,862319	0,818841	0,876812	0,891304	0,847826	0,859420
0,47	0,862319	0,840580	0,876812	0,884058	0,847826	0,862319
0,48 − 0,54	0,869565	0,840580	0,876812	0,891304	0,847826	0,865217
0,55	0,869565	0,847826	0,862319	0,876812	0,862319	0,863768
0,56 − 0,57	0,869565	0,855072	0,862319	0,876812	0,862319	0,865217
0,58	0,869565	0,855072	0,862319	0,876812	0,855072	0,863768
0,59	0,869565	0,855072	0,862319	0,869565	0,855072	0,862319
0,6 − 0,61	0,869565	0,862319	0,862319	0,869565	0,855072	0,863768
0,62	0,862319	0,862319	0,869565	0,876812	0,869565	0,868116
0,63	0,862319	0,862319	0,869565	0,876812	0,862319	0,866667
0,64 − 0,68	0,862319	0,862319	0,869565	0,876812	0,869565	0,868116
0,69 − 0,74	0,862319	0,869565	0,869565	0,869565	0,862319	0,866667
0,75 − 0,76	0,862319	0,869565	0,869565	0,876812	0,862319	0,868116
0,77	0,862319	0,862319	0,869565	0,876812	0,862319	0,866667
0,78	0,869565	0,862319	0,869565	0,876812	0,862319	0,868116
0,79 − 0,81	0,869565	0,869565	0,869565	0,876812	0,862319	0,869565
0,82 − 0,83	0,869565	0,869565	0,869565	0,884058	0,862319	0,871014
0,84	0,876812	0,876812	0,869565	0,884058	0,862319	0,873913
0,85 − 0,93	0,876812	0,869565	0,869565	0,884058	0,862319	0,872464
0,94 − 0,99	0,884058	0,876812	0,862319	0,869565	0,862319	0,871015
1	0,884058	0,876812	0,862319	0,862319	0,862319	0,869565

5 Conclusions

Results of tests show that the approach presented here yields results which in their optimal values are fully comparable with the best results obtained by other rough set based methods. The best results are obtained in last part (sect. 4) with weighted voting modified by using an additional parameter ε. This indicates that the possibility of improving the results further may lie in modifying granules by additional parameters at the cost of greater computational complexity. One has to underline the reduction in size of training sets as well as rule sets which are brought for by granulating data. Applications to this fact will be studied elsewhere.

Acknowledgements

This work is dedicated as homage to the memory of Professor Zdzisław Pawlak. The first author does encompass in his thoughts on the occasion of this work people who contributed essentially to the development of the idea of Professor Pawlak about nature of uncertainty.

References

1. Artiemjew, P.: On classification of data by means of rough mereological granules of objects and rules. In: Wang, G., Li, T., Grzymala-Busse, J.W., Miao, D., Skowron, A., Yao, Y. (eds.) RSKT 2008. LNCS, vol. 5009, pp. 221–228. Springer, Heidelberg (2008)
2. Artiemjew, P.: Classifiers from granulated data sets: Concept dependent and layered granulation. In: Proceedings RSKD 2007. Workshop at ECML/PKDD 2007, pp. 1–9. Warsaw Univ. Press, Warsaw (2007)
3. Bazan, J.G.: A comparison of dynamic and non–dynamic rough set methods for extracting laws from decision tables. In: Polkowski, L., Skowron, A. (eds.) Rough Sets in Knowledge Discovery, vol. 1, pp. 321–365. Physica Verlag, Heidelberg (1998)
4. Hájek, P.: Metamathematics of Fuzzy Logic. Kluwer, Dordrecht (1998)
5. Hoa, N.S.: Regularity analysis and its applications in Data Mining. In: Polkowski, L., Tsumoto, S., Lin, T.Y. (eds.) Rough Set Methods and Applications, pp. 289–378. Physica Verlag, Heidelberg (2000)
6. Pawlak, Z.: Rough sets. Int. J. Computer and Information Sci. 11, 341–356 (1982)
7. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning about Data. Kluwer, Dordrecht (1991)
8. Polkowski, L.: Rough Sets. Mathematical Foundations. Physica Verlag, Heidelberg (2002)
9. Poincare, H.: Science and Hypothesis. Walter Scott Publ., London (1905)
10. Polkowski, L.: Rough Sets. Mathematical Foundations. Physica Verlag, Heidelberg (2002)
11. Polkowski, L.: A rough set paradigm for unifying rough set theory and fuzzy set theory (a plenary lecture). In: Wang, G., Liu, Q., Yao, Y., Skowron, A. (eds.) RSFDGrC 2003. LNCS, vol. 2639, pp. 70–78. Springer, Heidelberg (2003); also Fundamenta Informaticae 54, 67–88 (2003)

12. Polkowski, L.: Toward rough set foundations. Mereological approach (a plenary lecture). In: Tsumoto, S., Słowiński, R., Komorowski, J., Grzymała-Busse, J.W. (eds.) RSCTC 2004. LNCS, vol. 3066, pp. 8–25. Springer, Heidelberg (2004)
13. Polkowski, L.: Formal granular calculi based on rough inclusions (a feature talk). In: [24], pp. 57–62
14. Polkowski, L.: A model of granular computing with applications (a feature talk). In: [25], pp. 9–16
15. Polkowski, L.: The paradigm of granular rough computing. In: Proceedings ICCI 2007. 6th IEEE Intern. Conf. on Cognitive Informatics, Lake Tahoe NV, USA, August 2007, pp. 145–163. IEEE Computer Society, Los Alamitos (2007)
16. Polkowski, L.: Granulation of knowledge in decision systems: The approach based on rough inclusions. The method and its applications. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS, vol. 4585, pp. 271–279. Springer, Heidelberg (2007)
17. Polkowski, L.: A Unified approach to granulation of knowledge and granular computing based on rough mereology: A Survey. In: Pedrycz, W., Skowron, A., Kreinovich, V. (eds.) Handbook of Granular Computing, ch. 16. John Wiley, NY (2008)
18. Polkowski, L.: On the idea of using granular rough mereological structures in classification of data. In: Wang, G., Li, T., Grzymala-Busse, J.W., Miao, D., Skowron, A., Yao, Y. (eds.) RSKT 2008. LNCS, vol. 5009, pp. 213–220. Springer, Heidelberg (2008)
19. Polkowski, L.: Rough mereology as a tool in analysis of vagueness. In: Wang, G., Li, T., Grzymala-Busse, J.W., Miao, D., Skowron, A., Yao, Y. (eds.) RSKT 2008. LNCS, vol. 5009, pp. 205–212. Springer, Heidelberg (2008)
20. Polkowski, L., Artiemjew, P.: On granular rough computing: Factoring clasifiers through granular structures. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS, vol. 4585, pp. 280–289. Springer, Heidelberg (2007)
21. Polkowski, L., Artiemjew, P.: A study in granular computing: On classifiers induced from granular reflections of data. In: Peters, J.F., et al. (eds.) Transactions on Rough Sets. LNCS, vol. 5390. Springer, Heidelberg (2008)
22. Polkowski, L., Skowron, A.: Rough mereology. In: Ras, Z., Zemankova, M. (eds.) ISMIS 1994. LNCS (LNAI), vol. 869, pp. 85–94. Springer, Heidelberg (1994)
23. Polkowski, L., Skowron, A.: Rough mereology: a new paradigm for approximate reasoning. International Journal of Approximate Reasoning 15(4), 333–365 (1997)
24. Proceedings of IEEE 2005 Conference on Granular Computing, GrC 2005, Beijing, China, July 2005. IEEE Press, Los Alamitos (2005)
25. Proceedings of IEEE 2006 Conference on Granular Computing, GrC 2006, Atlanta, USA, May 2006. IEEE Press, Los Alamitos (2006)
26. Skowron, A. et al.: RSES, `http://logic.mimuw.edu.pl/~rses/`
27. UCI Repository, `http://www.ics.uci.edu/~mlearn/databases/`
28. Wróblewski, J.: Adaptive aspects of combining approximation spaces. In: Pal, S.K., Polkowski, L., Skowron, A. (eds.) Rough – Neural Computing. Techniques for Computing with Words, pp. 139–156. Springer, Berlin (2004)

The Neurophysiological Bases of Cognitive Computation Using Rough Set Theory

Andrzej W. Przybyszewski

Department of Neurology, University of Massachusetts Medical Center, Worcester, MA US
and
Department of Psychology, McGill University, Montreal, Canada
Andrzej.Przybyszewski@umassmed.edu
http://www.umassmed.edu/neurology/faculty/Przybyszewski.cfm

Abstract. A popular view is that the brain works in a similar way to a digital computer or a Universal Turing Machine by processing symbols. Psychophysical experiments and our amazing capability to recognize complex objects (like faces) in different light and context conditions argue against symbolic representation and suggest that concept representation related to similarities may be a more appropriate model of brain function. In present work, by looking into anatomical and neurophysiological basis of how we classify objects shapes, we propose to describe computational properties of the brain by rough set theory (Pawlak, 1992 [1]). Concepts representing objects physical properties in variable environment are weak (not precise), but psychophysical space shows precise object categorizations. We estimate brain expertise in classifications of the object's components by analyzing single cell responses in the area responsible for simple shape recognition ([2]). Our model is based on the receptive field properties of neurons in different visual areas: thalamus, V1 and V4 and on feedforward (FF) and feedback (FB) interactions between them. The FF pathways combine properties extracted in each area into a vast number of hypothetical objects by using "driver logical rules", in contrast to "modulator logical rules" of the FB pathways. The FB pathways function may help to change weak concepts of objects physical properties into their crisp classification in psychophysical space.

Keywords: imprecise computation, bottom-up, top-down processes, neuronal activity.

1 Introduction

Humans can effortlessly recognize objects as complex as faces even if they have never seen them in a particular context before. We are perceptually insensitive to the exact properties of an objects part, but the same parts in different configurations or contexts may result in opposite effects, much like the Thatcher effect. Psychophysical experiments related to complex object and faces categorization show that object recognition is based on incomplete information about an objects parts.

One of the most popular models based on psychophysical experiments is the geon model related to the Recognition-by-components (RBC) theory [3]. A geon can be structurally described (GSD) as a two-dimensional representation of an arrangement of parts, each specified in terms of its non-accidental characterization and the relations amongst these parts [3]. Across objects, the parts (geons, or geometric icons) can differ in their nonaccidental properties (NAP). NAP are properties that do not change with small depth rotations of an object. The presence or absence of NAP of some geons or the different relations between them is the basis for discrimination of viewpoint invariant objects [3]. Consequently, complex objects can be described by a simple "alphabet" by utilizing a small set of structural primitive geons. However, RBC theory not only does not attempt to describe a complex, real scene by an alphabet of geons, but it is also incomplete, failing to distinguish many real objects.

Such experiments suggest that in every day life we not only perform object classifications based solely on partial information about an object, but that we also make accessible information about variations in that object's parts, such as its rotation or our viewpoint, indiscernible. An exact, crisp description (a set of values) of all the attributes of an object is therefore not only impossible because of the limitations of our visual system (such as a small area of sharp image, short perception time associated with the eye fixation period (see below), etc.), but also because we would like to identify the same object under different light conditions, contexts, localizations, rotations, etc. These difficulties in the context of our amazing capabilities in face and facial expression recognition led us (Prof. Zdzislaw Pawlak, Prof. Lech Polkowski, Prof. Andrzej Skowron, and myself) to discuss the application of rough set theory in order to find logical rules for complex object categorizations (the face project). Below I will present my summary regarding several points of our discussion.

As mentioned above, psychophysical and neurophysiological limitations led us to conclude that the brain-as-an-expert in complex object recognition may use vague concepts to process approximate information about a perceived object. Pawlak [1] proposed characterizing these concepts by their upper and lower approximations. The difference between the upper and lower approximations of a set of objects with related attributes is called its boundary region. Using the above characterizations, the concept of vagueness can be precisely expressed by the boundary region of a set. If a boundary region of a set is empty, the upper and lower approximations of the set are equal and the set is crisp (classical set theory). In this case, we can clearly classify the object on the basis of its properties as recognizable (a member of the set; logical value=1) or not recognizable (not a member of the set; logical value=0).

Crisp set also represents the classical approach to psychophysical experiments where near the limit of our vision, a subject can sometimes see the object and sometimes cannot. On the basis of averaging experimental results, we say that if the chance of seeing the object is over 50 % then the object is visible, and if it is below 50 %, then the object is not visible. The spectrum of an objects visibility is reduced to two values of a logical state: visible or not visible.

If the boundary region of a set is not empty, however, the set is rough and one can estimate the degree to which an object belongs to a set, or in other words, the precision with which the object is recognized. I claim that there are neurophysiological mechanisms in the brain responsible for shrinking the boundary regions along different dimensions in various visual areas of the brain. I describe these mechanisms with an example of the hierarchical structure: from the thalamus to area V4, the part of visual system responsible for simple shape discrimination. The thalamus classification characterizes an object by parts and questions how accurate divisions are using the rough inclusion relation or the rule that determines whether points are part of an object (Lesniewski mereology). For example, in order to find an object's contours, the surround portion of the LGNs receptive field should be outside of the object (see below for a more detailed analysis). In other visual areas such as V1, objects are partly described by oriented lines. More detailed measurements (see below) of area V1 receptive fields show many deviations from their sensitivity to only a single oriented edge. Their size for example, increases when light intensity decreases and it also depends on the mapping stimulus (i.e. dots vs. grating). Area V4 classifies objects to an extent as simple shapes coded in its neurons. For example, the mean size of the receptive field in area V4 is about 6 deg that means that there is partial overlap between an objects size and a single cell receptive field (RF) in area V4. Therefore, the relationship between object and receptive field in most situations is not crisp.

The famous mathematician Lukasiewicz's once asked whether or not it was true that "Jan will be in Warsaw next year?" This question initiated a course of research related to uncertainty using multi-valued logic. The activities of our sensory and motor systems are also related to uncertainty. In sensory-related motion, our eyes are constantly moving, with brief periods of fixation during which reliable information about interesting objects must be obtained and depending on the information received, a decision about the next eye movement must be made. The brain is continually verifying sensory information with predictions that are related to assumptions about the environment and possible object properties. Similarly, in a constantly changing environment, the brain has to make calculations and predictions with regards to behavior about how to meet or catch an object with uncertain properties and movement trajectories. These predictions are verified and corrected during each movement. We can therefore paraphrase Lukasiewicz's sentence in relation to subconsciously-made brain decisions as: "If I move my eyes to the right in the next 100 ms, will I know that it is Jan's face?" or "If I correct my hand trajectory in the next 50 ms will I catch the ball?"

The main purpose of this paper is to relate anatomical and neurophysiological brain properties to object categorization. In order to quantify the neurophysiological data and model perception, we will use rough set theory [1]. The complexity of the brain and many different methods of measurement generated huge amounts of data that led to unclear and contradictory theories. This work therefore compares data from the literature with data from our electrophysiological experiments using rough set theory and multi-valued logic. Using these precise descriptions, we would like to find closer connections between the electrophysiological and AI (expert) systems, as well as describe our results using psychological language.

Like Pawlak [1], we define an information system as $S = (U, A)$, where U is a set of objects and A is set of attributes. If $a \in A$, $u \in U$ the value $a(u)$ is a unique element of V_a (a value set of the attribute a). In agreement with the Leibniz principle we assume that objects are completely determined by their set of properties, meaning that attributes map objects to a set of attributes values. The indiscernibility relation of any subset B of A, or $IND(B)$, is defined [1] as the equivalence relation whose elements are the sets $u : b(u) = v$ as v varies in V_b, and $[u]_B$ - the equivalence class of u form *B-elementary granule*. The concept $X \subseteq U$ is *B-definable* if for each $u \in U$ either $[u]_B \subseteq X$ or $[u]_B \subseteq U - X$. $\underline{B}\, X = \{u \in U : [u]_B \subseteq X\}$ is a lower approximation of X. The concept $X \subseteq U$ is *B-indefinable* if exists such $u \in U$ that $[u]_B \cap X \neq \emptyset$. $\bar{B}X = \{u \in U : [u]_B \cap X \neq \emptyset\}$ is an upper approximation of X. The set $BN_B(X) = \bar{B}X - \underline{B}X$ will be referred to as the *B-boundary region* of X. If the boundary region of X is the empty set then X is exact (crisp) with respect to B; otherwise if $BN_B(X) \neq \emptyset$ X is not exact (rough) with respect to B.

In this paper, the universe U is a set of simple visual patterns that we used in our experiments [2, 4, 5], and which can be divided into equivalent indiscernibility classes related to their physically measured, computer generated attributes or *B-elementary granules*, where $B \subseteq A$.

The purpose of our research is to find how these objects are classified in the brain. We will therefore modify, after [1], the definition of the information system as $S = (U, C, D)$ where C and D are condition and decision attributes respectively. Decision attributes will classify elementary granules in accordance with neurological responses from a specific area of the visual brain. From a cognitive perspective, the percept of the object is classified into different categories in different visual areas, leading to different decisions (actions). The information system is equivalent to the decision table in which each object $u \in U$ is characterized by a series of the condition attributes and one decision attribute. The information system can be also seen as agents' intelligence where condition attributes described agents' percepts and decision attributes are related to the agents' action [6].

This work investigates the responses of cells in the thalamus that will divide all equivalent pattern classes into LGN-elementary granules $[u]_{LGN}$, as well as the responses of cells in area V1 that will divide all patterns into V1-elementary granules $[u]_{V1}$, and the responses of cells in area V4 that will divide all patterns into V4-elementary granules $[u]_{V4}$. All neurons in these areas are sensitive to certain attributes of the stimulus, such as space localization, but each area also performs distinct pattern classification. In consequence, one *B-elementary granule* will be classified in many different ways by neurons in different areas. All these granules: $[u]_{LGN}$, $[u]_{V1}$, $[u]_{V4}$ are exact. They cover the visual field in a unique way for a fixed eye position, even if the receptive fields of different cells overlap. Relationships between granules from different areas are rough, however, meaning that granules containing information related to feedforward connections $[u]_{LGN} \subseteq [u]_{V1} \subseteq [u]_{V4}$ and to feedback pathways: $[u]_{V4} \subseteq U_{s \in S}[u]^{s}_{V1} \subseteq U_{s \in S}[u]^{s}_{LGN}$ are rough inclusions, where $U_{s \in S}$ is mereological sum of all granules covering area of the the V4 neuron. As we will show below, each pathway obeys different logical rules. Our hypothesis is that the brain uses a hierarchical multi-level classification in order to find different important invariances at each level. These invariances may help to classify different presentations of the same object that, in different conditions, may lack or show changes in some of its parts.

Our model describes neurophysiological data in rough set theory [1] language and suggests that in order to classify complex objects the brain uses multi-valued logic, granular knowledge and rough mereology.

2 Basic Concepts

2.1 Objects' (stimuli) Attributes and Classification of the Brain Responses

We will represent our experimental data [2] in the following tables (Tabs. 1-5). In the first column are neural measurements. Neurons are identified using numbers related to a collection of figures in [2] concatenated with the cell number. Additional letters $(a, b, ...)$ denotes different measurements of the same cell. For example, $11a$ denotes the first measurement of a neuron numbered 1 Fig. 1, 11b the second measurement, etc. Simple stimuli properties are as characterized as follows: Most of our analysis will be related to data from Pollen et al. [2].

1. Orientation in degrees appears in the column labeled o, and orientation bandwidth is ob.
2. spatial frequency is denoted as sf, spatial frequency bandwidth is sfb
3. *x-axis* position is denoted by xp and the range of x-positions is xpr
4. *y-axis* position is denoted by yp and the range of y-positions is ypr
5. *x-axis* stimulus size is denoted by xs
6. *y-axis* stimulus size is denoted by ys
7. stimulus shape is denoted by s, values of s are following: for grating $s = 1$, for vertical bar $s = 2$, for horizontal bar $s = 3$, for disc $s = 4$, for annulus $s = 5$, for two stimuli $s = 22$ - two vertical bars, etc.

Stimulus attributes can be express as:

$$B = \{o, ob, sf, sfb, xp, xpr, yp, ypr, xs, ys, s\}.$$

Generally, we divide all cell responses into n ranges, but in this paper, for simplicity, we use three ranges of the neural responses. Activity below the threshold in between *10 and 20 spikes/s* is defined as a *category 0* cell response. Activity above the threshold is defined as *category 1*, and activity above *30 - 40 spikes/s* as *category 2*. We analyze only the dominant component of the cell response, which in LGN and simple V1 cells is *linear* (the first harmonic F_1) and in the complex V1 cell and in V4 cell is *nonlinear* (related to F_0 or the mean changes in the neuronal discharges). The reason for choosing the minimum significant cell activity of *10 - 20 spikes/s* is as follows: during normal activity our eyes are constantly moving. The fixation periods are between 100 and $300ms$, similar to those of monkeys. Assuming that a single neuron, in order to give reliable information about an object, must fire a minimum of *2-3 spikes* during the eye fixation period, we obtain a minimum frequency of *10 to 20 spikes/s.* We assume that these discharges are determined by the bottom-up information (hypothesis testing) and that they are related to the sensory information about an object's form. The brain is constantly making predictions, which are verified by comparing them with sensory information. These tests are performed in a positive feedback loop [4, 7]. If prediction is in agreement with the hypothesis, we assume that activity of the cell increases approximately twofold similarly to the strength of the feedback from V1 to LGN [4]. This increased activity is related to category 2. Cell responses (r) are divided into 3 ranges:

category 0: activity below the threshold *10 - 20 sp/s* labeled by r_0;
category 1: activity above the threshold labeled by r_1;
category 2: activity above 30 - 40sp/s labeled by r_2.

2.2 Logic of the Anatomical Connections

As it was mentioned above, our model consists of three interconnected visual areas. Their connections can be divided into feedforward (FF) and feedback (FB) pathways. We have proposed [7] that the role of the FF pathways is to test the hypothesis about stimulus attributes and the function of the FB pathways is to make predictions. Below, we suggest that the different anatomical properties of the FB and FF pathways may determine their different logical rules. We define LGN_i, as LGN *i-cell* attributes for cells $i = 1, \dots, n$, $V1_j$ as primary visual cortex *j-cell* attributes for cells $j = 1, \dots, m$, and $V4_k$ as area V4 attributes for cells $k = 1, \dots, l$. The specific stimulus attributes for a single cell can be found in the neurophysiological experiment by recording cell responses to the set of various test stimuli. As we have mentioned above, cell responses are divided into several (here 3) ranges, which will define several granules for each cell. It is different from the classical receptive field definition, which assumes that the cell responds (*logical value 1*) or does not respond (*logical value 0*) to the stimulus with certain attributes. In the classical electrophysiological approach all receptive field granules are crisp. In our approach, cell responses below the threshold (r_0), have logical value 0, whereas the maximum cell responses (r_2), have a logical value 1. We will introduce cell responses between r_0 and r_2, in this paper there is only one value, r_1. The physiological interpretation of cell responses between the threshold and the maximum response may be related to the influence of the feedback or horizontal pathways. We assume that the tuning of each structure is different and we will look for decision rules in each level that give responses r_1 and r_2. For example, we assume that r_1 means that the local structure is tuned to the attributes of the stimulus and such granule for *j cell* in area V1 will be define as $[u]_{1V1j}$.

Decision Rules for a single neuron. Each neuron in the central nervous system sums up its synaptic inputs as a postsynaptic excitatory (*EPSPs*) and/or inhibitory (*IPSPs*) potentials that may cause its membrane potential to exceed the threshold and to generate an action potential. A single neuron approximates collective (thousands of interacting synapses with different weights) input information to the distributive one (unique decision in a single output). In principle, a single spike (action potential) can be seen as a decision of the neuron, but in this work we will not take into account internal dynamics of the system and therefore we will estimate neuronal activity as spikes mean frequency (as described above). This complex synaptic potential summation process is related in sensory (here only visual) systems with the receptive field properties of each neuron. Below we will show how neurons in different parts of the brain change visual information in their receptive fields into decisions.

Decision Rules for LGN. Each LGN cell is sensitive to luminance changes in a small part of the visual field called the receptive field (RF). The cells in the LGN have the concentric center-surround shapes of their RFs, which are similar to that in the retinal ganglion cells [8]. We will consider only on- and off type RFs. The on - (off) type cells increase (decrease) their activity by an increase of the light luminance in their receptive field center and/or decrease of the light luminance in the RF surround (Fig. 1). Below are examples of the decision rules for on-, and off-type LGN cells with their RF positions: xp_0, yp_0. We assume that there is no positive feedback from higher areas therefore their maximum responses are r_1.

DR_LGN_1:

$$xp_0 \wedge yp_0 \wedge xs_{0.1} \wedge ys_{0.1} \wedge s_4 \to r_1 \tag{1}$$

DR_LGN_2:

$$xp_0 \wedge yp_0 \wedge xs_{0.3} \wedge ys_{0.3} \wedge s_5 \to r_1 \tag{2}$$

which we interpret that the changes in the luminance of the light spot s_4 that covers the RF center (the first rule) or annulus s_5 that covers the RF surround (the second rule) gives neuronal response r_1. We assume that other stimulus parameters like contrast, speed and frequency of luminance changes, etc. are constant and optimal, and that the cell is linear and therefore we measure response of the cell activity synchronized with the stimulus changes (the first harmonic). Depending on the cell type the phase shift between stimulus and the response is near 0 or $180deg$ if we do not take into account the phase shift related to the response delay. Instead, using light spots or annuli one can use a single, modulated with the drifting grating, circular patch covering the classical RF. By changing the spatial frequency of the drifting grating one can stimulate only the RF center for high spatial frequencies or center and surround for lower spatial frequencies, which gives the following decision rule:

DR_LGN_3:

$$xp_0 \wedge yp_0 \wedge xs_{0.3} \wedge ys_{0.3} \wedge sf_{0.4} \rightarrow r_1 \tag{3}$$

where for example: $sf = 0.4c/d$ stimulates RF center and surround, $sf \geq 1c/d$ stimulates RF center only. Notice that in agreement with the above rules eqs. (1-3) LGN cells do not differentiate between light spot, light annulus, and patch modulated with grating. All these different objects represent the same LGN-elementary granule.

Decision Rules for area V1. In the primary visual cortex (area V1) neurons obtain a new property: sensitivity to the stimulus orientation, which is not observed in lower areas: retina or LGN [9]. The area V1 has at least two different cell types: simple and complex. They can be characterized by spatial relationships between their incremental (on) and decremental (off) subfields. In a simple cell on and off subfields are seperated, whereas a complex cell is characterized by the overlap of its subfields. In consequence simple cells are linear (the first harmonic dominates in their responses: $F1/F0 > 1$), whereas complex cells are nonlinear ($F1/F0 < 1$). The classical V1 RF properties can be found using small flashing light spots, moving white or dark bars or gratings. We will give an example of the decision rules for the RF mapped with the moving white and dark bars [5]. A moving white bar gives the following decision rule:

DR_V1_1:

$$xp_i \wedge yp_0 \wedge xs_k \wedge ys_1 \wedge s_2 \rightarrow r_1 \tag{4}$$

The decision rule for a moving dark bar is given as:

DR_V1_2:

$$xp_j \wedge yp_0 \wedge xs_l \wedge ys_1 \wedge s_2 \rightarrow r_1 \tag{5}$$

where xp_i is the x-position of the incremental subfield, where xp_j is the x-position of the decremental subfield, yp_0 is the y-position of the both subfields, xs_k, xs_l, ys_1 are horizontal and vertical sizes of the RF subfields, and s_2 is a vertical bar which means that this cell is tuned to the vertical orientation. We have skipped other stimulus attributes like movement velocity, direction, amplitude, etc. For simplicity we assume that the cell is not direction sensitive, it gives the same responses to both direction of bar movement and to the dark and light bars and that cell responses are symmetric around the x middle position (xp). An overlap index [10] is defined as:

$$OI = \frac{0.5(xs_k + xs_l) - |xp_i - xp_j|}{0.5(xs_k + xs_l) + |xp_i - xp_j|}$$

OI compares sizes of increment (xs_k) and decrement (xs_l) subfields to their separation ($|xp_i - xp_j|$). After [11], if $OI \leq 0.3$ ("non-overlapping" subfields) it is the simple cell with dominating first harmonic response (linear) and r_1 is the amplitude of the first harmonic. If $OI \geq 0.5$ (overlapping subfields), it is the complex cell with dominating $F0$ response (nonlinear) and r_1 are changes in the mean cell activity. Hubel and Wiesel [9] have proposed that the complex cell RF is created by convergence of several simple cells in a similar way like V1 RF properties are related to RF of LGN cells (Fig. 1). However, there is recent experimental evidence that the nonlinearity of the complex cell RF may be related to the feedback or horizontal connections [12].

Decision Rules for area V4. The properties of the RFs in area V4 are more complex than that in area V1 or in the LGN and in most cases they are nonlinear. It is not clear what exactly optimal stimuli for cells in V4 are, but a popular hypothesis states that the V4 cells code the simple, robust shapes. Below there is an example from [13] of the decision rules for a narrow (0.4 deg) and long (4 deg) horizontal or vertical bars placed in different positions of area V4 RF:

DR_V4_1:

$$o_0 \wedge ypr_m \wedge (yp_{-2.2} \vee yp_{0.15}) \wedge xs_4 \wedge ys_{0.4} \rightarrow r_2 \tag{6}$$

DR_V4_2:

$$o_{90} \wedge xpr_m \wedge (xp_{-0.6} \vee xp_{1.3}) \wedge xs_{0.4} \wedge ys_4 \rightarrow r_1 \tag{7}$$

The first rule relates area V4 cell responses to a moving horizontal bar (o_0) and the stimulus in the second rule is a moving vertical bar (o_{90}), ypr_m, xpr_m have meaning of the tolerance for the y or x bar positions (more details in the Result section). The horizontal bar placed narrowly in two different y-positions ($yp_{-2.2}, yp_{0.15}$) gives strong cell responses (**DR_V4_1**), and the vertical bar placed with wide range in two different x-positions ($xp_{-0.6}, xp_{1.3}$) gives medium cell responses.

Decision Rules for feedforward connections from LGN → V1. Thalamic axons target specific cells in layers 4 and 6 of the primary visual cortex (V1). Generally we assume that there is a linear summation of LGN cells (approximately $10 - 100$ of them [14]) to one V1 cell. It was proposed [9] that the LGN cells determine the orientation of the V1 cell in the following way: LGN cells which have a direct synaptic connection to V1 neurons have their receptive fields arranged along a straight line on the retina (Fig. 1). In this Hubel and Wiesel [9] classical model the major assumption is that activity of all (four in Fig. 1) LGN cells is necessary for a V1 cell to be sensitive to the specific stimulus (oriented light bar). This principle determines syntax of the LGN to V1 decision rule, by using logical and meaning that if one LGN cell does not respond then there is no V1 cell response. After Sherman and Guillery [15] we will call such inputs drivers. Alonso et al. [14] showed that there is a high specificity between RF properties of the LGN cells which have monosynaptic connections to a V1 simple cell. This precision goes beyond simple retinotopy and includes such RF properties as RF sign, timing, subregion strength and sign [14]. The decision rule for the feedforward LGN to V1 connections are following:

DR_LGN_V1_1:

$$r_1^{LGN}(x_i, y_i) \wedge r_1^{LGN}(x_{i+1}, y_i) \wedge \ldots \wedge r_1^{LGN}(x_{i+n}, y_i) \rightarrow r_1^{V1} \tag{8}$$

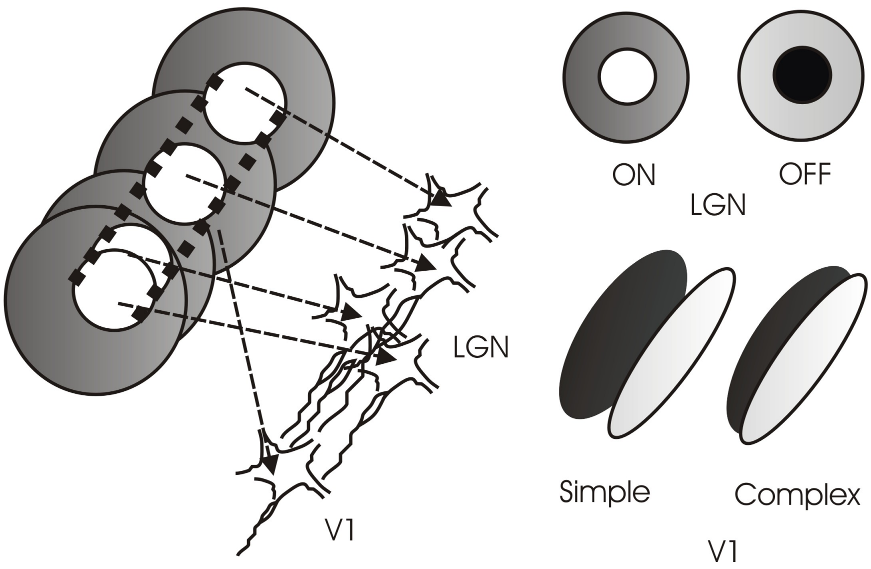


Fig. 1. On the left: modified schematic of the model proposed by [9]. Four LGN cells with circular receptive fields arranged along a straight line on the retina have direct synaptic connection to V1 neuron. This V1 neuron is orientation sensitive as marked by the thick, interrupted lines. On the right: receptive fields of two types of LGN cells, and two types of area V1 cells.

DR_LGN_V1_2:

$$r_1^{LGN}(x_i, y_i) \wedge r_1^{LGN}(x_{i+1}, y_{i+1}) \wedge \ldots \wedge r_1^{LGN}(x_{i+n}, y_{i+1}) \rightarrow r_1^{V1} \qquad (9)$$

where the first rule determines response of cells in V1 with optimal horizontal orientation, and the second rule says that the optimal orientation is 45 degrees; (x_i, y_i) is the localization of the RF in x-y Euclidian coordinates of the visual field. Notice that these rules assume that V1 RF is completely determined by the FF pathway from the LGN.

Decision Rules for feedback connections from V1→LGN. There are several papers showing the existence of the feedback connections from V1 to LGN [16-20]. In [20], authors have quantitatively compared the visuotopic extent of geniculate feedforward afferents to V1 with the size of the RF center and surround of neurons in V1 input layers and the visuotopic extent of V1 feedback connections to the LGN with the RF size of cells in V1. Area V1 feedback connections restrict their influence to LGN regions visuotopically coextensive with the size of the classical RF of V1 layer 6 cells and commensurate with the LGN region from which they receive feedforward connections. In agreement with [15] we will denote feedback inputs modulators with following decision rules:

DR_V1_LGN_1:

$$(r_1^{V1} \vee r_1^{LGN}(x_i, y_i)), (r_1^{V1} \vee r_1^{LGN}(x_i, y_{i+1}), (r_1^{V1} \vee r_1^{LGN}(x_{i+1}, y_{i+1})), \ldots$$
$$\ldots, r_1^{LGN}(x_{i+2n}, y_{i+2n})) \rightarrow r_2^{LGN} \tag{10}$$

This rule says that when the activity of a particular V1 cell is in agreement with activity in some LGN cells their responses increase from r_1 to r_2, and $r_1^{LGN}(x_i, y_i)$ means r_1 response of LGN cell with coordination (x_i, y_i) in the visual field, and r_2^{LGN} means r_2 response of all LGN cells in the decision rules which activity was coincidental with the feedback excitation, it is a pattern of LGN cells activity.

Decision Rules for feedforward connections V1 → V4. There are relatively small direct connections from V1 to V4 bypassing area V2 [20], but we also must take into account V1 to V2 [21] and V2 to V4 connections, which are highly organized but variable, especially in V4 [22] feedforward connections. We simplify that V2 has similar properties to V1 but have a larger size of the RF. We assume that, like from the retina to LGN and from LGN to V1 direct or indirect connections from V1 to V4 provide driver input and fulfill the following decision rules:

DR_V1_V4_1:

$$r_1^{V1}(x_i, y_i) \wedge r_1^{V1}(x_{i+1}, y_i) \wedge \ldots \wedge r_1^{V1}(x_{i+n}, y_i) \rightarrow r_1^{V4} \tag{11}$$

DR_V1_V4_2:

$$r_1^{V1}(x_i, y_i) \wedge r_1^{V1}(x_{i+1}, y_{i+j}) \wedge \ldots \wedge r_1^{V1}(x_{i+n}, y_{i+m}) \rightarrow r_1^{V4} \tag{12}$$

We assume that, the RF in area V4 sums up driver inputs from regions in the areas V1and V2 of cells with highly specific RF properties, not only retinotopically correlated.

Decision Rules for feedback connections from V4→V1. Anterograde anatomical tracing [23] has shown axons backprojecting from area V4 directly to area V1 or sometimes with branches in area V2. Axons of V4 cells span in area V1 in large territories with most terminations in layer 1, which can be either distinct clusters or in linear arrays. These specific for each axon branches determine decision rules that will have similar syntax (see below) but anatomical structure of the particular axon may introduce different semantics. Their anatomical structures maybe related to the specific receptive field properties of different V4 cells. Distinct clusters may have terminals on V1 cells near pinwheel centers (cells with different orientations arranged radially), whereas a linear array of terminals may be connected to V1 neurons with similar orientation preference. In consequence, some parts of the V4 RF would have preference for certain orientations and others may have preference for the certain locations but be more flexible to different orientations. This hypothesis is supported by recent intracellular recordings from neurons located near pinwheels centers which, in contrast to other narrowly tuned neurons, showed subthreshold responses to all orientations [24]. However, neurons which have fixed orientation can change other properties of their receptive field like for example spatial frequency, therefore the feedback from area V4 can tune them to expected spatial details in the RF (M. Sur, Brenda Milner Symposium, 22 Sept. 2008, MNI McGill University, Montreal).

The V4 input modulates V1 cell in the following way:

DR_V4_V1_1:

$$(r_1^{V4} \vee r_1^{V1}(x_i, y_i)), (r_1^{V4} \vee r_1^{V1}(x_i, y_{i+1}), (r_1^{V4} \vee r_1^{V1}(x_{i+1}, y_{i+1})), \ldots$$

$$\ldots, (r_1^{V4} \vee r_1^{V1}(x_{i+n}, y_{i+m})) \rightarrow r_2^{V1} \tag{13}$$

Meaning of $r_1^{V1}(x_i, y_i)$ and r_2^{V1} are same as explained above for the V1 to LGN decision rule.

Decision Rules for feedback connections V4→LGN. Anterograde tracing from area V4 showed axons projecting to different layers of LGN and some of them also to the pulvinar [25] These axons have widespread terminal fields with branches non-uniformly spread about several millimeters (Fig. 2). Like descending axons in V1, axons from area V4 have within their LGN terminations, distinct clusters or linear branches (Fig. 2). These clusters and branches are characteristic for different axons and as it was mentioned above their differences may be related to different semantics in the decision rule below:

DR_V4_LGN_1:

$$(r_1^{V4} \vee r_1^{LGN}(x_i, y_i)), (r_1^{V4} \vee r_1^{LGN}(x_i, y_{i+1}), (r_1^{V4} \vee r_1^{LGN}(x_{i+1}, y_{i+1})), \ldots$$

$$\ldots, (r_1^{V4} \vee r_1^{LGN}(x_{i+n}, y_{i+m})) \rightarrow r_2^{LGN} \tag{14}$$

Meaning of $r_1^{LGN}(x_i, y_i)$ and r_2^{LGN} are same as explained above for the V1 to LGN decision rule.

Notice that interaction between FF and FB pathways extends a classical view that the brain as computer uses two-valued logic. This effect in psychophysics can be paraphrased as: “I see it but it does not fit my predictions”. In neurophysiology, we assume that a substructure could be optimally tuned to the stimulus but its activity does not fit to the FB predictions. Such interaction can be interpreted as the third logical value. If there is no stimulus, the response in the local structure should have a logical value 0, if stimulus is optimal for the local structure, it should have logical value $\frac{1}{2}$, and if it also is tuned to expectations of higher areas (positive feedback) then response should have logical value 1. Generally it becomes more complicated if we consider many interacting areas, but in this work we use only three-valued logic.

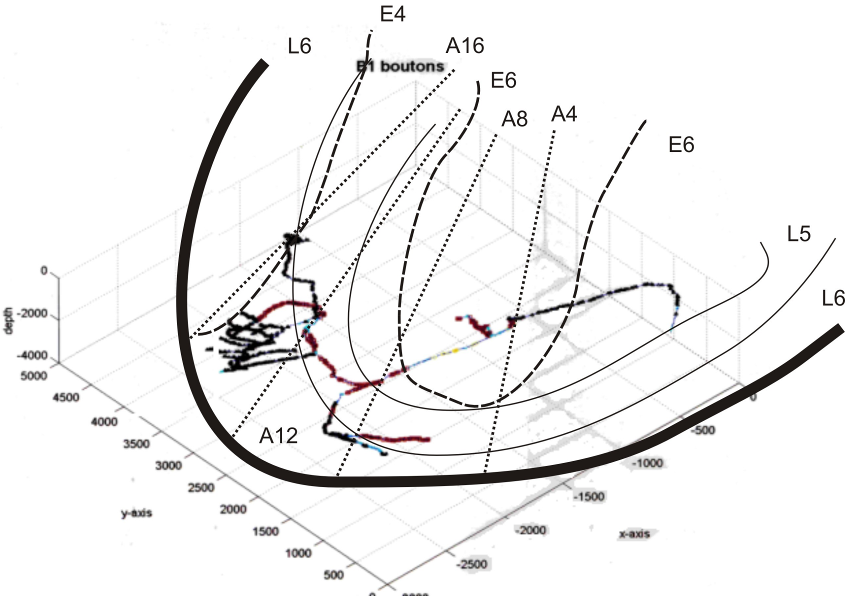


Fig. 2. Boutons of the descending axon from area V4 with terminals in different parvocellular layers of LGN: layer 6 in black, layer 5 in red, layer 4 in yellow. Total number of boutons for this and other axons was between 1150 and 2075. We estimated that it means that each descending V4 axon connects to approximately 500 to over 1000 LGN (mostly parvocellular) cells [25]. Thick lines outline LGN; thin lines shows layers 5 and 6, dotted line azimuth, and dashed lines show elevation of the visual field covered by the descending axon. This axon arborization extension has approximately V4 RF size.

3 Results

We have used our model as a basis for an analysis of the experimental data from the neurons recorded in the monkey's area V4 [2]. In [2], it was shown that the RF in V4 can be divided into several subfield that, stimulated separately, can give us the first approximation of the concept of the shape to which the cell is tuned [13]. We have also shown that subfields are tuned to stimuli with similar orientation [2]. In Fig. 3, we demonstrate that the receptive field subfields have not only similar preferred orientations but also spatial frequencies [2]. We have divided cell responses into three categories (see Methods) by horizontal lines in plots A-D of Fig. 3.

We have draw a line near spike frequency *17 spikes/s*, which separates responses of category r_1 (above) from r_0 (below the threshold line). Horizontal lines plotted near spike frequency *34 spikes/s* separate responses of category r_2 (above) from r_1 (below). The stimulus attributes related to these three response categories were extracted in the decision table (Table 1). We summarize results of our analysis in Figs. 3H and G from Table 1. Fig. 3H presents a schematic of a possible stimulus that would give medium cell responses (r_1). One can imagine

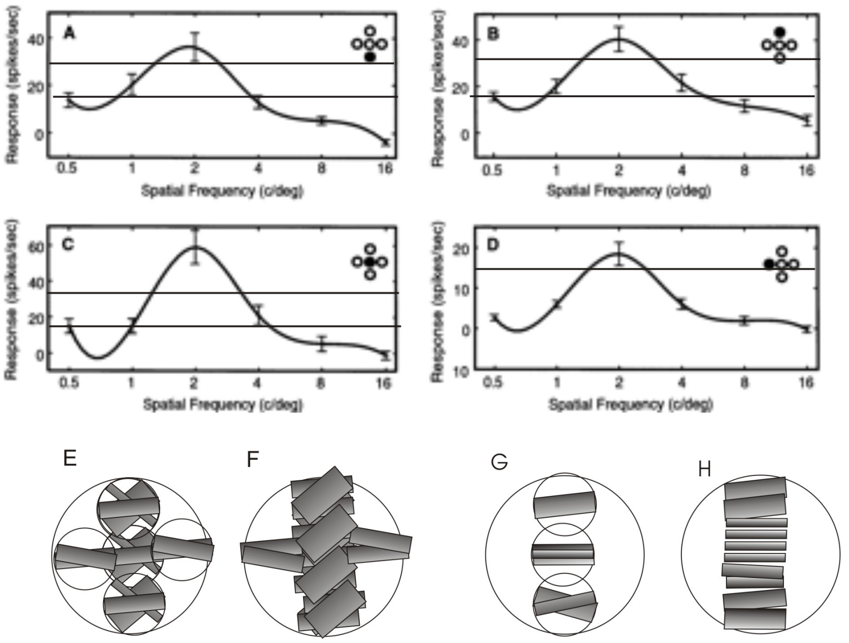


Fig. 3. Modified plots from [2]. Curves represent responses of V4 neurons to their RF subfields grating stimulations with different spatial frequencies (SF). (A-D) SF selectivity curves across RF with positions indicated in insets. The centers of tested subfields were 2 deg apart. (E-H) Schematic representation summarizing orientation and SF selectivity of subfields presented in A-D and in [2]. These figures are based on the decision table 1, for stimuli in E, F cell responses were r_1, for stimuli in G, H cell responses were r_2, (F) and (G) represent a possible stimulus configuration from schematics (E) and (F).

several classes of possible stimuli assuming that subfield responses will sum up linearly (for example see Fig. 3F). Fig. 3G shows a schematic of a possible stimulus set-up, which would give r_2 response that as we have assumed, is related not only to the local but also the global visual cortex tuning. One can notice that in the last case only subfields in the vertical row give strong independent responses (Fig. 3H).

We assign the narrow (ob_n), medium (ob_m), and wide (ob_w) orientation bandwidth as follows: ob_n if ($ob : 0 < ob < 50deg$), medium ob_m if ($ob : 50deg < ob < 100deg$), wide ob_w if ($ob : ob > 100deg$). We assign the narrow (sfb_n), medium (sfb_m), and wide (sfb_w) spatial frequency bandwidth: sfb_n if ($sfb : 0 < sfb < 2c/deg$), medium sfb_m if ($sfb : 2c/deg < sfb < 2.5c/deg$), wide sfb_w if ($sfb : sfb > 2.5c/deg$). For simplicity in the following decision rules, we assume that the subfields are not direction sensitive; therefore responses to stimulus orientation *0* and *180 deg* should be same.

Table 1. Decision table for one cell responses to subfields stimulation Fig. 3C-F and Fig.5 in [2]. Attributes $xpr, ypr, sf = 2c/deg, s$ are constant and they are not presented in the table. Cells 3* are from Fig. 3 in [2] and cells 5* are from Fig. 5 in [2] processed in Fig. 3.

cell	o	ob	sfb	xp	yp	r
3c	172	105	0	0	0	1
3c1	10	140	0	0	0	1
3c2	180	20	0	0	0	2
3d	172	105	0	0	-2	1
3d1	5	100	0	0	-2	1
3d2	180	50	0	0	-2	2
3e	180	0	0	-2	0	0
3f	170	100	0	0	2	1
3f1	10	140	0	0	2	1
3f2	333	16	0	0	2	2
5a	180	0	3	0	-2	1
5a1	180	0	0.9	0	-2	2
5b	180	0	3.2	0	2	1
5b1	180	0	1	0	2	2
5c	180	0	3	0	0	1
5c1	180	0	1.9	0	0	2
5d	180	0	0.8	0	0	1

Our results from the separate subfields stimulation study can be presented as the following decision rules:

DR_V4_3:

$$o_{180} \wedge sf_2 \wedge ((ob_w \wedge sfb_w \wedge xp_0 \wedge (yp_{-2} \vee yp_0 \vee yp_2)))\vee$$
$$\vee (ob_n \wedge sfb_n \wedge yp_0 \wedge (xp_{-2} \vee xp_2)) \rightarrow r_1 \tag{15}$$

DR_V4_4:

$$o_{180} \wedge sf_2 \wedge ob_n \wedge sfb_n \wedge xp_0 \wedge (yp_{-2} \vee yp_0 \vee yp_2) \rightarrow r_2 \tag{16}$$

These decision rules can be interpreted as follows: disc shaped grating stimuli with wide bandwidths of orientations or spatial frequencies when placed along vertical axis of the receptive field evoke medium cell responses. However, similar discs when placed horizontally to the left or to the right from the middle of the RF, must have narrow orientation and spatial frequency to evoke medium cell responses. Only a narrowly tuned disc in spatial frequency and orientation placed vertically from the middle of the receptive field can evoke strong cell responses. Notice that Figs 3F and 3H show possible configurations of the optimal stimulus. This approach is similar to the assumption that an image of the object is initially represented in terms of the activation of a spatially arrayed set of multiscale, multioriented detectors like arrangements of simple cells in V1 (metric templates in subordinate-level object classification of Lades et al. [26]). However, this approach does not take into account interactions between several stimuli, when more than one subfield is stimulated, and we will show below there is a strong nonlinear interaction between subfields. We analyzed experiments where the RF is stimulated at first with a single small vertical bar and later with two bars changing their horizontal positions. One example of V4 cell responses to thin (*0.25 deg*) vertical bars in different horizontal positions is shown in the upper left part of Fig. 4 (Fig. 4E). Cell response has maximum amplitude for the middle ($XPos = 0$) bar position along the $x - axis$. Cell responses are not symmetrical around 0. In Fig. 2F, the same cell (cell 61 in table 2) is tested with two bars. The first bar stays at the 0 position, while the second bar changes its position along $x - axis$. Cell responses show several maxima dividing the receptive field into four areas. However, this is not always the case as responses to two bars in another cell (cell 62 in table 2) show only two minima (Fig. 2G). Horizontal lines in plots of both figures divide cell responses into the three categories r_0, r_1, r_2, which are related to the mean response frequency (see Methods). Stimuli attributes and cell responses classified into categories are shown in table 2 for cells in Fig. 4 and in table 3 for cells in Fig. 5.

We assign the narrow (xpr_n), medium (xpr_m), and wide (xpr_w) x position ranges as follows: xpr_n if (xpr : $0 < xpr \leq 0.6$), medium xpr_m if (xpr : $0.6 < xpr \leq 1.2$), wide xpr_w if (xpr : $xpr > 1.2$). We assign the narrow (ypr_n), medium (ypr_m), and wide (ypr_w) y position range: ypr_n if (ypr : $0 < ypr \leq 1.2$), medium ypr_m if (ypr : $1.2 < xpr \leq 1.6$), wide ypr_w if (ypr : $ypr > 1.6$).

On the basis of Fig. 3 and the decision table 2 (also compare with [18]) the one-bar study can be presented as the following decision rules:

DR_V4_5:

$$o_{90} \wedge xpr_n \wedge xp_{0.1} \wedge xs_{0.25} \wedge ys_{0.4} \rightarrow r_2 \tag{17}$$

DR_V4_6:

$$o_{90} \wedge xpr_w \wedge xp_{-0.2} \wedge xs_{0.25} \wedge ys_{0.4} \rightarrow r_1 \tag{18}$$

We interpret these rules that r_1 response in eq. (18) does not effectively involve the feedback to the lower areas: V1 and LGN. The descending V4 axons have excitatory synapses not only on relay cells in LGN and pyramidal cells in V1, but also on inhibitory interneurons in LGN and inhibitory double banquet cells in layer 2/3 of V1. As an effect of the feedback, only a narrow range of area V4 RF responded with a high r_2 activity to a single bar stimulus, whereas in the outside area excitatory and inhibitory feedback influences compensated each other.

On the basis of Fig. 4 the decision table, the two-bar horizontal interaction study can be presented as the following **Two-bar Decision Rules (DRT):**

DRT_V4_1:

$$o_{90} \wedge xpr_n \wedge ((xp_{-1.9} \vee xp_{0.1} \vee xp_{1.5}) \wedge xs_{0.25} \wedge ys_{0.4})_1 \wedge (o_{90} \wedge xp_0 \wedge xs_{0.25} \wedge ys_{0.4})_0 \rightarrow r_2 \tag{19}$$

DRT_V4_2:

$$o_{90} \wedge xpr_m \wedge ((xp_{-1.8} \vee xp_{-0.4} \vee xp_{0.4} \vee xp_{1.2}) \wedge xs_{0.25} \wedge ys_{0.4})_1 \wedge$$
$$\wedge (o_{90} \wedge xp_0 \wedge xs_{0.25} \wedge ys_{0.4})_0 \rightarrow r_1 \qquad (20)$$

One-bar decision rules can be interpreted as follows: the narrow vertical bar evokes a strong response in the central positions, and medium responses in a larger area near the central position. Two-bar decision rules claim that: the cell responses to two bars are strong if one bar is in the middle of the RF (bar with index 0 in decision rules) and the second narrow bar (bar with index 1 in decision rules) is in the certain, specific positions in the RF eq. (19). But when the second bar is in less precise positions, cell responses became weaker eq. (20). Responses of other cells are sensitive to other bar positions (Fig. 4G). These differences could be correlated with anatomical variability of the descending

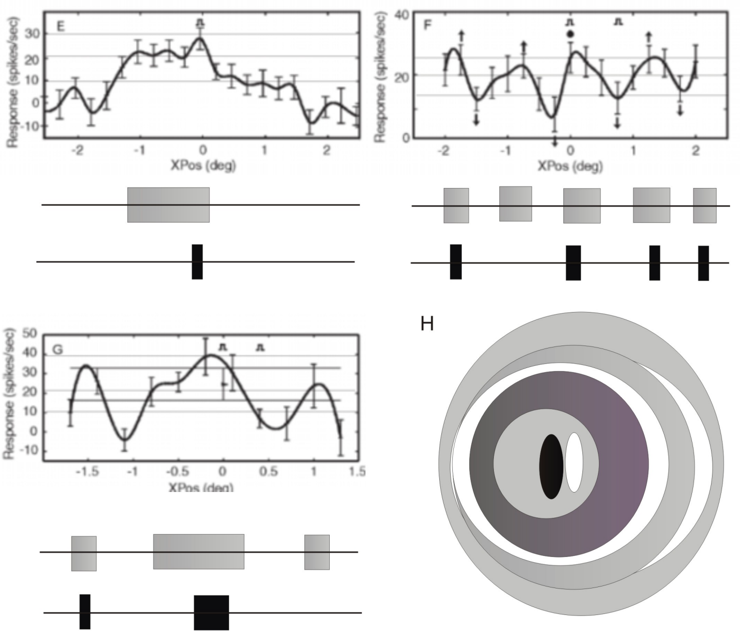


Fig. 4. Modified plots from [2]. Curves represent responses of several cells from area V4 to small single (E) and double (F, G) vertical bars. Bars change their position along x-axis (Xpos). Responses are measured in spikes/sec. Mean cell responses ± SE are marked in E, F, and G. Cell responses are divided into three ranges by thin horizontal lines. Below each plot are schematics showing bar positions giving r_1 (gray) and r_2 (black) responses; below (E) for a single bar, below (F and G) for double bars (one bar was always in position 0). (H) This schematic extends responses for horizontally placed bars (E) to the whole RF: white color shows excitatory, black color inhibitory interactions between bars. Bars' interactions are asymmetric in the RF.

Table 2. Decision table for cells shown in Fig. 4. Attributes *o, ob, sf, sfb* were constant and are not presented in the table.

cell	*xp*	*xpr*	*xs*	*ys*	*s*	*r*
61e	-0.7	1.4	0.25	4	2	1
61f1	-1.9	0.2	0.25	4	22	2
61f2	0.1	0.2	0.25	4	22	2
61f3	1.5	0.1	0.25	4	22	2
61f4	-1.8	0.6	0.25	4	12	1
61f5	-0.4	0.8	0.25	4	22	1
61f6	0.4	0.8	0.2 5	4	22	1
61f7	1.2	0.8	0.25	4	22	1
62g1	-1.5	0.1	0.25	4	22	2
62g2	-0.15	0.5	0.25	4	22	2
62g3	-1.5	0.6	0.25	4	22	1
62g4	-0.25	1.3	0.25	4	22	1
62g5	1	0.6	0.25	4	22	1
63h1	-0.5	0	0.5	1	44	2
63h2	1	1	1	1	44	1
63h3	0.2	0.1	0.25	4	22	2

Table 3. Decision table for one cell shown in Fig. 5. Attributes yp, ypr are constant and are not presented in the table. We introduce another parameter of the stimulus, difference in the direction of drifting grating of two patches: ddg = 0 when drifting are in the same directions, and ddg = 1 if drifting in two patches are in opposite directions.

cell	*xp*	*xpr*	*xs*	*ys*	*ddg*	*r*
64c	-4.5	3	1	1	1	2
64c1	-1.75	1.5	1	1	1	1
64c2	-0.5	1	1	1	1	2
64d	-6	0	1	8	0	2
64d1	-3.5	4.8	1	8	0	1

axons connections. As mentioned above, V4 axons in V1 have distinct clusters or linear branches. Descending pathways are modulators, which means that they follow the logical "or" rule. This rule states that cells in area V1 become more active as a result of the feedback only if their patterns "fit" to the area V4 cell "expectation".

The decision table (Table 3) based on Fig. 5 describes cell responses to two patches placed in different positions along x-axis of the receptive field (RF). Figure 5 shows that adding the second patch reduced single patch cell responses. We have assumed that cell response to a single patch placed in the middle of the RF is r_2. The second patch suppresses cell responses to a greater extent when it is more similar to the first patch (Fig. 5D).

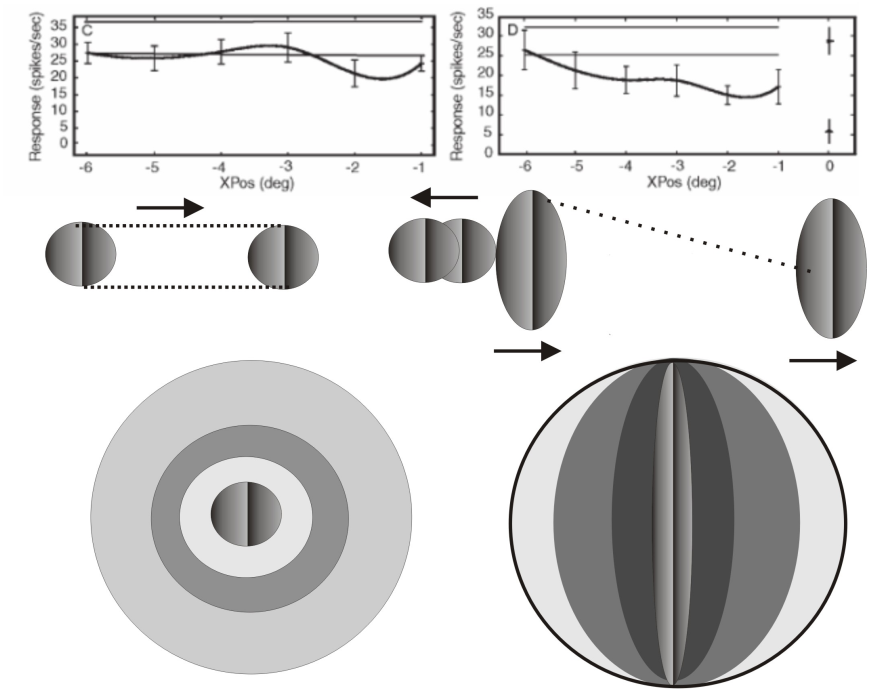


Fig. 5. Modified plots from [2]. Curves represent V4 cell responses to two patches with gratings moving in opposite direction - patch 1 deg diameter (C) and in the same (D) directions for patch 1 deg wide and 8 deg long. One patch is always at x-axis position 0 and the second patch changes its position as it is marked in *XPos* coordinates. The horizontal lines represent 95% confidence intervals for the response to a single patch in position 0. Below C and D, schematics show the positions of the patches and their influences on cell responses. Arrows are showing the direction of moving gratings. The lower part of the figure shows two schematics of the excitatory (white) and inhibitory (black) interactions between patches in the RF. Patches with gratings moving in the same directions (right schematic) show larger inhibitory areas (more dark color) than patches moving in opposite directions (left schematic).

Two-patch horizontal interaction decision rules are as follows:

DRT_V4_3:

$$ddg_1 \wedge (o_0 \wedge xpr_3 \wedge xp_{4.5} \wedge xs_1 \wedge ys_1)_1 \wedge (o_0 \wedge xp_0 \wedge xs_1 \wedge ys_1)_0 \rightarrow r_2 \quad (21)$$

DRT_V4_4:

$$ddg_1 \wedge (o_0 \wedge xpr_1 \wedge xp_{0.5} \wedge xs_1 \wedge ys_1)_1 \wedge (o_0 \wedge xp_0 \wedge xs_1 \wedge ys_1)_0 \rightarrow r_2 \quad (22)$$

DRT_V4_5:

$$ddg_0 \wedge (o_0 \wedge xpr_{4.8} \wedge xp_{3.5} \wedge xs_1 \wedge ys_8)_1 \wedge (o_0 \wedge xp_0 \wedge xs_1 \wedge ys1)_0 \rightarrow r_1 \quad (23)$$

Table 4. Decision table for cells in Fig. 6. Attributes $yp, ypr, xs = ys = 0.5deg, s = 33$ (two discs) are constant and are not presented in the table. We introduce another parameter of the stimulus, difference in polarities of two patches: $dp = 0$ if polarities are same, and $dp = 1$ if polarities are opposite.

cell	*xp*	*xpr*	*dp*	*r*
81a	-0.1	0.5	0	1
81a1	-1.75	0.3	0	1
81a2	-1.2	1	1	1
81a3	1.25	1.5	1	1
81a4	-1.3	0.3	1	2
81a5	-1.3	0.3	1	2
81a6	1.5	0.4	1	2
81b	-1.4	0.6	1	1
81b1	0.9	0.8	1	1
81b2	0.9	0.2	1	2

These decision rules can be interpreted as follows: patches with drifting in opposite directions gratings give strong responses when positioned very near (overlapping) or 150% of their width apart one from the other eqs. (21, 22). Interaction of patches with a similar grating evoked small responses in large extend of the RF eq. (23).

Generally, interactions between similar stimuli evoke stronger and more extended inhibition than between different stimuli. These and other examples can be generalized to other classes of objects.

Two-spot horizontal interaction decision rules are as follows:

DRT_V4_6:

$$dp_0 \wedge s_{33} \wedge (((xp_{-0.1} \wedge xpr_{0.5}) \vee (xp_{-1.75} \wedge xpr_{0.3})) \wedge xs_{0.5})_1 \wedge (xp_0 \wedge xs_{0.5})_0 \rightarrow r_1 \tag{24}$$

DRT_V4_7:

$$dp_1 \wedge s_{33} \wedge (((xp_{-1.2} \wedge xpr_1) \vee (xp_{1.25} \wedge xpr_{1.5})) \wedge xs_{0.5})_1 \wedge (xp_0 \wedge xs_{0.5})_0 \rightarrow r_1 \tag{25}$$

DRT_V4_8:

$$dp_1 \wedge s_{33} \wedge (((xp_{-1.3} \wedge xpr_{0.2}) \vee (xp_{1.5} \wedge xpr_{0.4})) \wedge xs_{0.5})_1 \wedge (xp_0 \wedge xs_{0.5})_0 \rightarrow r_2 \tag{26}$$

DRT_V4_9:

$$dp_1 \wedge s_{33} \wedge (((xp_{-1.4} \wedge xpr_{0.6}) \vee (xp_{0.9} \wedge xpr_{0.8})) \wedge xs_{0.5})_1 \wedge (xp_0 \wedge xs_{0.5})_0 \rightarrow r_1 \tag{27}$$

DRT_V4_10:

$$dp_1 \wedge s_{33} \wedge ((xp_{0.9} \wedge xpr_{0.2}) \wedge xs_{0.5})_1 \wedge (xp_0 \wedge xs_{0.5})_0 \rightarrow r_2 \tag{28}$$

where dp is the difference in light polarities between two light spots (s_{33}), and subscript 1 is related to spot changing its x-axis position, whereas subscript 0 is related to the spot in 0 position on x-axis.

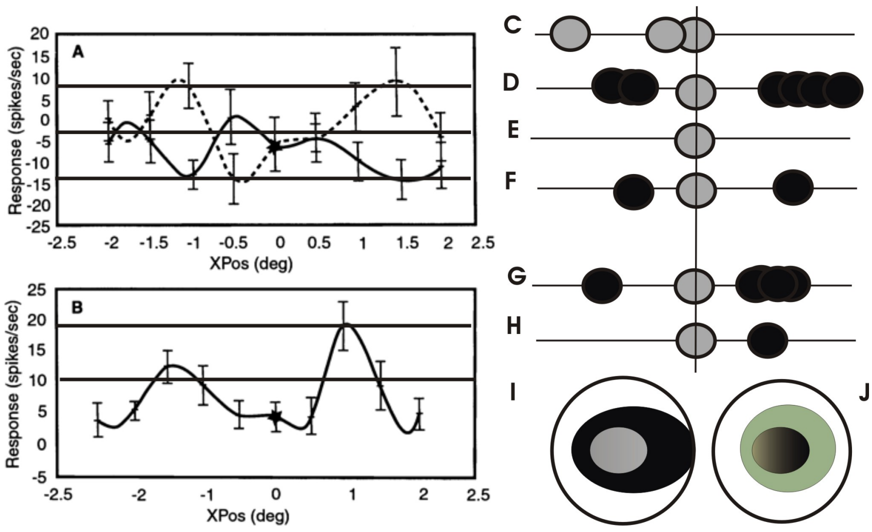


Fig. 6. Modified plots from [2]. Curves represent V4 cell responses to pair of 0.5 deg diameter bright and dark discs tested along width axis. Continuous lines mark the curves for responses to different polarity stimuli, and the same polarity stimuli are marked by dashed line. Schematics for cell responses showed in (A) are in (C-F) and (I, J). Schematics for cell responses in (B) are in (G) and (H). Interactions between same polarity light spots (C) are different than interactions between different polarities patches (D-H). Small responses (*class 1*) are in (C), (D), (G), and larger responses (*class 2*) are in (E), (F), (H). (E) shows that there is no r_2 responses in same polarity two spots interactions. (I) shows small excitatory (gray) in a short range and strong inhibitory (black) interactions between same polarity spots and (J) shows short range inhibitory (dark) and longer range excitatory interactions between different polarities spots.

We propose the following classes of the object's **Parts Interaction Rules:**

PIR1: Facilitation when stimulus consists of multiple similar thin bars with small distances (about 0.5 deg) between them, and suppression when the distance between bars is larger than 0.5 deg. Suppression/facilitation is very often a nonlinear function of the distance. In our experiments (Fig. 3), cell responses to two bars were periodic along the receptive field with dominating periods of about 30, 50, or 70% of the RF width. These nonlinear interactions were also observed along vertical and diagonals of the RF and often show strong asymmetries in relationship to the RF middle.

PIR2: Strong inhibition when stimulus consists of multiple similar patches filled with gratings with the distance between patch edges ranging from 0 deg (touching) to 2 deg, weak inhibition when distance is between 3 to 5 deg through the RF width.

PIR3: If bars or patches have different attributes like polarity or drifting directions, their suppression is smaller and localized facilitation at the small distance between stimuli is present. As in bar interaction, suppression/facilitations between patches or bright/dark discs can be periodic along different RF axis and often asymmetric in the RF.

We have tested the above rules in nine cells from area V4 by using discs or annuli filled stimuli with optimally oriented and variable in spatial frequencies drifting gratings (Pollen et al. [2] Figs. 9, 10). Our assumptions were that if it is a strong inhibitory mechanism as described in the rule PRI2 then responses to annulus with at least 2 deg inner diameters will be stronger than responses to the disc. In addition by changing spatial frequencies of gratings inside the annulus, we have expected eventually to find other periodicities along the RF width as described by PIR3.

In summary, we wanted to find out what relations there are between stimulus properties and area V4 cell responses or whether B-elementary granules have equivalence classes of the relation $IND\{r\}$ or V4-elementary granules, or whether $[u]_B \Rightarrow [u]_{B4}$. It was evident from the beginning that because different area V4 cells have different properties, their responses to the same stimuli will be different, therefore we wanted to know if the rough set theory will help us in our data modeling.

We assign the spatial frequency: low (sf_l), medium (sf_m), and high (sf_h) as follows: sf_l if ($sf : 0 < sf \leq 1c/deg$), medium sf_m if ($sf : 1c/deg < sf \leq 4c/deg$), high sf_h if ($sf : sf > 4c/deg$). On the basis of this definition we calculate for each row in Table 5 the spatial frequency range by taking into account the spatial frequency bandwidth (sf_b). Therefore 107a is divided to 107al and 107am, 108a to 108al and 108am, and 108b to 108bl, 108bm, and 108bh.

Stimuli used in these experiments can be placed in the following ten categories:

$$Y_0 = |sf_l\ xo_7\ xi_0\ s_4| = \{101, 105\}$$

$$Y_1 = |sf_l\ xo_7\ xi_2\ s_5| = \{101a, 105a\}$$

$$Y_2 = |sf_l\ xo_8\ xi_0\ s_4| = \{102, 104\}$$

$$Y_3 = |sf_l\ xo_8\ xi_3\ s_5| = \{102a, 104a\}$$

$$Y_4 = |sf_l\ xo_6\ xi_0\ s_4| = \{103, 106, 107, 108, 20a, 20b\}$$

$$Y_5 = |sf_l\ xo_6\ xi_2\ s_5| = \{103a, 106a, 107al, 108bl\}$$

$$Y_6 = |sf_l\ xo_4\ xi_0\ s_4| = \{108al\}$$

$$Y_7 = |sf_m\ xo_6\ xi_2\ s_5| = \{107am, 108bm\}$$

$$Y_8 = |sf_m\ xo_4\ xi_0\ s_4| = \{107a, 108am\}$$

$$Y_9 = |sf_h\ xo_6\ xi_2\ s_5| = \{108bh\}$$

Table 5. Decision table for eight cells comparing the center-surround interaction. All stimuli were concentric, and therefore attributes were not xs, ys, but xo outer diameter, xi inner diameter. All stimuli were localized around the middle of the receptive field so that $xp = yp = xpr = ypr = 0$ were constant and we did not put them in the table. The optimal orientation were normalized, denoted as 1, and removed from the table.

cell	sf	sfb	xo	xi	s	r
101	0.5	0	7	0	4	0
101a	0.5	0	7	2	5	1
102	0.5	0	8	0	4	0
102a	0.5	0	8	3	5	0
103	0.5	0	6	0	4	0
103a	0.5	0	6	2	5	1
104	0.5	0	8	0	4	0
104a	0.5	0	8	3	5	2
105	0.5	0	7	0	4	0
105a	0.5	0	7	2	5	1
106	0.5	0	6	0	4	1
106a	0.5	0	6	3	5	2
107	0.5	0.25	6	0	4	2
107a	2.1	3.8	6	2	5	2
107b	2	0	4	0	4	1
108	0.5	0	6	0	4	1
108a	2	0	4	0	4	2
108b	5	9	6	2	5	2
20a	0.5	0	6	0	4	1
20b	0.5	0	6	0	4	2

These are equivalence classes for stimulus attributes, which means that in each class they are indiscernible $IND(B)$. We have normalized orientation bandwidth to 0 in $\{20a, 20b\}$ and spatial frequency bandwidth to 0, in cases $\{107, 107a, 108a, 108b\}$, and put values covered by the bandwidth to the spatial frequency parameters. There are three ranges of responses denoted as r_o, r_1, r_2. Therefore on the basis of the neurological data there are the following three categories of cell responses:

$$|r_o| = \{101, 102, 102a, 103, 104, 105\}$$

$$|r_1| = \{101a, 103a, 105a, 107b, 108, 20a\}$$

$$|r_2| = \{104a, 106a, 107, 107al, 107am, 108al, 108am, 108bl, 108bm, 108bh, 20b\}$$

which are denoted as X_o, X_1, X_2.

We will calculate the lower and upper approximation [1] of the brains basic concepts in term of stimulus basic categories:

$\underline{B}\, X_0 = Y_0 \cup Y_2 = \{101, 105, 102, 104\}$
$\bar{B}X_0{=}Y_0{\cup}Y_2{\cup}Y_3{\cup}Y_4{=}\{101, 105, 102, 104, 102a, 104a, 103, 106, 107, 108, 20a, 20b\}$
$\underline{B}\, X_1 = Y_1 = \{101a, 105a\}$
$\bar{B}X_1 = Y_1 \cup Y_5 \cup Y_6 \cup Y_4 =$
$\{101a, 105a, 103a, 107al, 108b, 106a, 20b, 107b, 108a, 103, 107, 106, 108, 20a\}$

$\underline{B} X_2 = Y_7 \cup Y_9 = \{107am, 108bm, 108bh\}$
$\bar{B}X_2 = Y_7 \cup Y_9 \cup Y_8 \cup Y_6 \cup Y_3 \cup Y_4 \cup Y_5 = \{107am, 108bm, 108bh, 107b, 108am, 102a, 104a, 103a, 107a, 108bl, 106a, 20b, 103, 107, 106, 108, 20a, 108al\}$

Concept 0 and concept 1 are roughly $B-defined$, which means that only with some approximation we have found that the stimuli do not evoke a response, or evoke weak or strong response in the area V4 cells. Certainly a stimulus such as Y_0 or Y_2 does not evoke a response in all our examples, in cells 101, 105, 102, 104. Also stimulus Y_1 evokes a weak response in all our examples: 101a, 105a. We are interested in stimuli that evoke strong responses because they are specific for area V4 cells. We find two such stimuli, Y_7 and Y_9. In the meantime other stimuli such as Y_3, Y_4 evoke no response, weak or strong responses in our data.

We can find the quality [1] of our experiments by comparing properly classified stimuli $POSB(r) = \{101, 101a, 105, 105a, 102, 104, 107am, 108bm, 108bh\}$ to all stimuli and to all responses: $\gamma\{r\} = \frac{card\{101,101a,105,105a,102,104,107am,108bm,108bh\}}{card\{101,101a,,20a,20b\}}$ $= 0.38$. We can also ask what percentage of cells we fully classified. We obtain consistent responses from 2 of 9 cells, which means that $\gamma = 0.22$. This is related to the fact that for some cells we have tested more than two stimuli. What is also important from an electrophysiological point of view is there are negative cases. There are many negative instances for the concept 0, which means that in many cases this brain area responds to our stimuli; however it seems that our concepts are still only roughly defined.

We have following decision rules:

DR_V4_7:

$$sf_l \wedge xo_7 \wedge xi_2 \wedge s_5 \to r_1 \tag{29}$$

DR_V4_8:

$$sf_l \wedge xo_7 \wedge xi_0 \wedge s_4 \to r_0 \tag{30}$$

DR_V4_9:

$$sf_l \wedge xo_8 \wedge xi_0 \wedge s_4 \to r_0 \tag{31}$$

DR_V4_10:

$$(sf_m \vee sf_h) \wedge xo_6 \wedge xi_2 \wedge s_5 \to r_2 \tag{32}$$

These can be interpreted as the statement that a large annulus (s_5) evokes a weak response, but a large disc (s_4) evokes no response when there is modulation with low spatial frequency gratings. However, somewhat smaller annulus containing medium or high spatial frequency objects evokes strong responses. It is unexpected that certain stimuli evoke inconsistent responses in different cells (Table 5):

103: $sf_l \wedge xo_6 \wedge xi_0 \wedge s_4 \to r_0$
106: $sf_l \wedge xo_6 \wedge xi_0 \wedge s_4 \to r_1$
107: $sf_l \wedge xo_6 \wedge xi_0 \wedge s_4 \to r_2$

A disc with not very large dimension containing a low spatial frequency grating can evoke no response (103), a small response (106), or a strong response (107).

4 Discussion

Physical properties of objects are different from their psychological representation. Grdenfors [27] proposed to describe the principle of human perceptual system as grouping objects by similarities in the conceptual space. Human perceptual systems group together similar objects with unsharp boundaries [27], which means that objects are related to their parts by rough inclusion or that different parts belong to objects with some approximation (degree) [28]. We suggest that similarity relations between objects and their parts are related to the hierarchical relationships between different visual areas. These similarities may be related to synchronizations of multi-resolution, parallel computations and are difficult to simulate using a digital computer [29].

Treisman [30] proposed that our brains extract features related to different objects using two different procedures: parallel and serial processing. The "basic features" were identified in psychophysical experiments as elementary features that can be extracted in parallel. Evidence of parallel features extraction comes from experiments showing that the extraction time becomes independent of the number of objects. Other features need serial searches, so that the extraction time is proportional to the number of objects. High-level serial processing is associated with integration and consolidation of items combined with conscious awareness. Other low-level parallel processes are rapid, global, related to high-efficiency categorization of items and largely unconscious [30]. Treisman [30] showed that instances of a disjunctive set of at least four basic features could be detected through parallel processing. Other researchers have provided evidence for parallel detection of more complex features, such as shape from shading [31] or experience-based learning of features of intermediate complexity [32].

Thorpe et al. [33] in recent experiments, however, found that human and non-human primates can rapidly and accurately categorize briefly flashed natural images. Human and monkey observers are very good at deciding whether or not a novel image contains an animal even when more than one image is presented simultaneously [34]. The underlying visual processing reflecting the decision that a target was present is under $150ms$ [33]. These findings are in contradiction to the classical view that only simple, "basic features", likely related to early visual areas like V1 and V2, are processed in parallel [30] Certainly, natural scenes contain more complex stimuli than "simple" geometric shapes. It seems that the conventional, two-stage perception-processing model needs correction, because to the "basic features" we must add a set of unknown intermediate features. We propose that at least some intermediate features are related to receptive field properties in area V4. Area V4 has been associated with shape processing because its neurons respond to shapes [35] and because lesions in this area disrupt shape discrimination, complex-grouping discriminations [36], multiple viewpoint shape discriminations [37], and rotated shape discriminations [38]. Area V4 responses are also driven by curvature or circularity, which was recently observed by mean of the human fMRI [39].

By applying rough sets to V4 neuron responses, we have differentiated between bottom-up information (hypothesis testing) related to the sensory input, and predictions, some of which can be learned but are generally related to positive feedback from higher areas. If a prediction is in agreement with a hypothesis, object classification will change from category 1 to category 2. Our research suggests that such decisions can be made very effectively during pre-attentive, parallel processing in multiple visual areas. In addition, we found that the decision rules of different neurons can be inconsistent.

One should take into account that modeling complex phenomena demands the use of local models (captured by local agents), if one would like to use the multiagent terminology [6]) that should be fused afterwards. This process involves negotiations between agents [6] to resolve contradictions and conflicts in local modeling. One of the possible approaches in developing methods for complex concept approximations can be based on the layered learning [41]. Inducing concept approximation should be developed hierarchically starting from concepts that can be directly approximated using sensor measurements toward complex target concepts related to perception. This general idea can be realized using additional domain knowledge represented in natural language.

We have proposed decision rules for different visual areas and for FF and FB connections between them. However in processing our V4 experimental data, we also have found inconsistent decision rules. These inconsistencies could help process different aspects of the properties of complex objects. The principle is similar to that observed in the orientation tuning cells of the primary visual cortex. Neurons in V1 with overlapping receptive fields show different preferred orientations. It is assumed that this overlap helps extract local orientations in different parts of an object. However, it is still not clear which cell will dominate if several cells with overlapping receptive fields are tuned to different attributes of a stimulus. Most models assume a "winner takes all" strategy; meaning that using a convergence (synaptic weighted averaging) mechanism, the most dominant cells will take control over other cells, and less represented features will be lost. This approach is equivalent to the two-valued logic implementation. Our finding from area V4 seems to support a different strategy than the "winner takes all" approach. It seems that different features are processed in parallel and then compared with the initial hypothesis in higher visual areas. We think that descending pathways play a major role in this verification process. At first, the activity of a single cell is compared with the feedback modulator by logical conjunction to avoid hallucinations. Next, the global, logical disjunction ("modulators") operation allows the brain to choose a preferred pattern from the activities of different cells. This process of choosing the right pattern may have strong anatomical basis because individual axons have variable and complex terminal shapes, facilitating some regions and features against other so called salient features (for example Fig. 2). Learning can probably modify the synaptic weights of the feedback boutons, fine-tuning the modulatory effects of feedback.

Neurons in area V4 integrate an object's attributes from the properties of its parts in two ways: (1) within the area via horizontal or intra-laminar local excitatory-inhibitory interactions, (2) between areas via feedback connections tuned to lower visual areas. Our research put more emphasis on feedback connections because they are probably faster than horizontal interactions [42]. Different neurons have different Part Interactions Rules (PIR as described in the Results section) and perceive objects by way of multiple "unsharp windows" (Figs. 4, 6). If an object's attributes fit the unsharp window, a neuron sends positive feedback [3] to lower areas, which as described above, use "modulator logical rules" to sharpen the attribute-extracting window and therefore change the neurons response from class 1 to class 2 (Fig. 4 J and K; Fig. 6 C to D, E to F, and G to H). The above analysis of our experimental data leads us to suggest that the central nervous system chiefly uses at least two different logical rules: "driver logical rule" and "modulator logical rule." The first, "driver logical rule," processes data using a large number of possible algorithms (over-representation). The second, "modulator logical rule," supervises decisions and chooses the right algorithm.

Below we will look at possible cognitive interpretations of our model using the shape categorization task as an example. The classification of different objects by their different attributes has been regarded as a single process termed "subordinate classification" [40]. Relevant perceptual information is related to subordinate-level shape classification by distinctive information of the object like its size, surface, curvature of contours, etc. There are two theoretical approaches regarding shape representation: metric templates and invariant parts models. As mentioned above, both theories assume that an image of the object is represented in terms of cell activation in areas like V1: a spatially arrayed set of multi-scale, multi-oriented detectors ("Gabor jets"). Metric templates [26] map object values directly onto units in an object layer, or onto hidden units, which can be trained to differentially activate or inhibit object units in the next layer [41]. Metric templates preserve the metrics of the input without the extraction of edges, viewpoint invariant properties, parts or the relations among parts. This model discriminates shape similarities and human psychophysical similarities of complex shapes or faces [25]. Matching a new image against those in the database is done by allowing the Gabor jets to independently change their own best fit (change their position). The similarities of two objects will be the sum of the correlations in corresponding jets. When this methods is used, changes in object or face position or changes in facial expressions can achieve 95% accuracy between several hundreds faces [43]. The main problems with the Lades model [26] described above are that it does not distinguish among the largest effects in object recognition it is insensitive to contour variations, which are very important psychophysically speaking, and it is insensitive to salient features (non-accidental properties [NAP]) [3].

The model we propose here suggests that these features are probably related to effects of feedback pathways, which may strengthen differences, signal salient features and also assemble other features, making it possible to extract contours. A geon structural description (GSD) is a two-dimensional representation of an arrangement of parts, each specified in terms of its non-accidental characterization and the relations amongst these parts [38]. Across objects, the parts (geons) can differ in their NAP. NAP are properties that do not change with

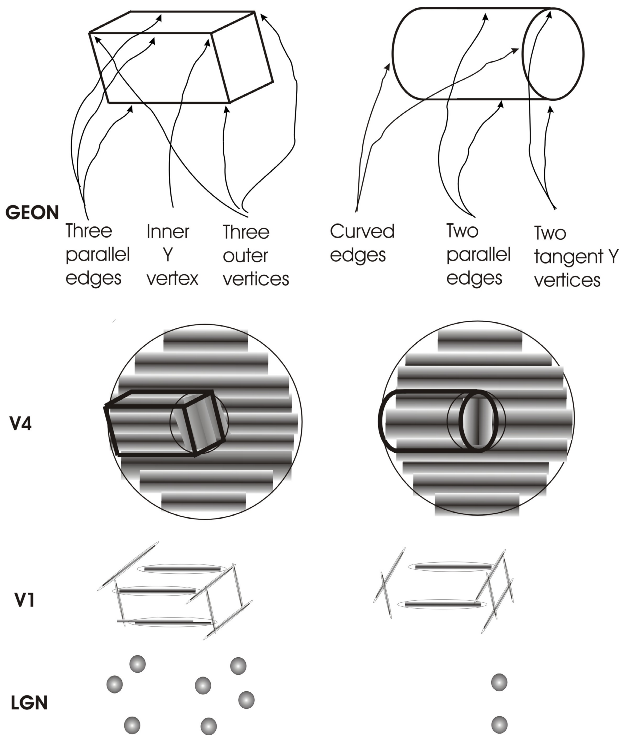


Fig. 7. Comparison of differences in nonaccidental properties between a brick and a cylinder using geon [3] and our model. The geon shows attributes from psychological space like curves, parallels or vertices, which may be different in different subjects. The neurological model compares properties of both objects on the basis of a single cell recordings from the visual system. Both objects can stimulate similar receptive fields in area V4. These receptive fields are sensitive in annuli - they extract orientation change in different parts of the RF [2]. Area V1 RFs are sensitive to edge orientations, whereas LGN RFs extract spots related to corners. All these different attributes are put together by FF and FB pathways.

small depth rotations of an object. The presence or absence of the NAP of some geons or the different relations between them may be the basis for subordinate level discrimination [38]. The advantage of the GSD is that the representation of objects in terms of their parts and the relations between them is accessible to cognition and fundamental for viewpoint invariant perception. Our neurological model introduces interactions between RF parts as in the geon model; however, our parts are defined differently than the somewhat subjective parts of the GSD model. Fig. 7 shows differences in a simple objects understanding between geon and our neurological approach. The top part of this figure shows differences in nonaccidental properties between a brick and a cylinder [3]. We propose hierarchical definition of parts based on neurophysiological recordings from the visual system. Both objects may be classified in V4 by the receptive field discriminating between different stimulus orientations in its central and peripheral parts as it is schematically presented in Fig. 7 [2]. Another, different classification is performed by area V1, where oriented edges are extracted from both objects (Fig. 7). However, even more precise classification is performed in LGN where objects are seen as sets of small circular shapes similar to receptive fields in the retina (bottom part of Fig. 7).

In our model, interactions between parts and NAPs are associated with the role of area V4 in visual discrimination, as described in the above lesion experiments [34-36]. However, feedback from area V4 to the LGN and area V1 could be responsible for the possible mechanism associated with the properties of the GSD model. The different interactions between parts may be related to the complexity and the individual shapes of different axons descending from V4. Their separated cluster terminals may be responsible for invariance related to small rotations (NAP). These are the anatomical bases of the GSD model, although we hypothesize that the electrophysiological properties of the descending pathways (FB), defined above as the modulator, are even more important. The modulating role of the FB is related to the anatomical properties of the descending pathways' logic. Through this logic, multiple patterns of the coincidental activity between the LGN or V1 and FB can be extracted. One may imagine that these differently extracted patterns of activity correlate with the multiple viewpoints or shape rotations defined as NAP in the GSD model.

In summary, by applying rough set theory to model neurophysiological data we have shown a new approach for objects categorization in psychophysical space. Two different logical rules are applied to indiscernibility classes of LGN, V1, and V4 receptive fields: "driver logical rules" put many possible objects' properties together and "modulator logical rules" choose these attributes which are in agreement with our previous experiences.

Acknowledgement. Thanks to Carmelo Milo for his technical help, as well to Farah Averill and Dana Hayward for their help in editing the manuscript.

References

1. Pawlak, Z.: Rough Sets - Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht (1991)
2. Pollen, D.A., Przybyszewski, A.W., Rubin, M.A., Foote, W.: Spatial receptive field organization of macaque V4 neurons. Cereb Cortex 12, 601–616 (2002)
3. Biederman, I.: Recognition-by-components: a theory of human image understanding. Psychol. Rev. 94(2), 115–147 (1987)
4. Przybyszewski, A.W., Gaska, J.P., Foote, W., Pollen, D.A.: Striate cortex increases contrast gain of macaque LGN neurons. Vis. Neurosci. 17, 485–494 (2000)
5. Przybyszewski, A.W., Kagan, I., Snodderly, M.: Eye position influences contrast responses in V1 of alert monkey [Abstract]. Journal of Vision 3(9), 698, 698a (2003), `http://journalofvision.org/3/9/698/`
6. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice Hall Series in Artificial Intelligence (2003)

7. Przybyszewski, A.W., Kon, M.A.: Synchronization-based model of the visual system supports recognition. Program No. 718.11. 2003 Abstract Viewer/Itinerary Planner. Society for Neuroscience, Washington, DC (2003)
8. Kuffler, S.W.: Neurons in the retina; organization, inhibition and excitation problems. Cold Spring Harb. Symp. Quant. Biol. 17, 281–292 (1952)
9. Hubel, D.H., Wiesel, T.N.: Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J. Physiol. 160, 106–154 (1962)
10. Schiller, P.H., Finlay, B.L., Volman, S.F.: Quantitative studies of single-cell properties in monkey striate cortex. I. Spatiotemporal organization of receptive fields. J. Neurophysiol. 39, 1288–1319 (1976)
11. Kagan, I., Gur, M., Snodderly, D.M.: Spatial organization of receptive fields of V1 neurons of alert monkeys: comparison with responses to gratings. J. Neurophysiol. 88, 2557–2574 (2002)
12. Bardy, C., Huang, J.Y., Wang, C., FitzGibbon, T., Dreher, B.: 'Simplification' of responses of complex cells in cat striate cortex: suppressive surrounds and 'feedback' inactivation. J. Physiol. 574, 731–750 (2006)
13. Przybyszewski, A.W.: Checking Brain Expertise Using Rough Set Theory. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 746–755. Springer, Heidelberg (2007)
14. Alonso, J.M., Usrey, W.M., Reid, R.C.: Rules of connectivity between geniculate cells and simple cells in cat primary visual cortex. J. Neurosci. 21(11), 4002–4015 (2001)
15. Sherman, S.M., Guillery, R.W.: The role of the thalamus in the flow of information to the cortex. Philos. Trans. R Soc. Lond. B Biol. Sci. 357(1428), 1695–1708 (2002)
16. Lund, J.S., Lund, R.D., Hendrickson, A.E., Bunt, A.H., Fuchs, A.F.: The origin of efferent pathways from the primary visual cortex, area 17, of the macaque monkey as shown by retrograde transport of horseradish peroxidase. J. Comp. Neurol. 164, 287–303 (1975)
17. Fitzpatrick, D., Usrey, W.M., Schofield, B.R., Einstein, G.: The sublaminar organization of corticogeniculate neurons in layer 6 of macaque striate cortex. Vis. Neurosci. 11, 307–315 (1994)
18. Ichida, J.M., Casagrande, V.A.: Organization of the feedback pathway from striate cortex (V1) to the lateral geniculate nucleus (LGN) in the owl monkey (Aotus trivirgatus). J. Comp. Neurol. 454, 272–283 (2002)
19. Angelucci, A., Sainsbury, K.: Contribution of feedforward thalamic afferents and corticogeniculate feedback to the spatial summation area of macaque V1 and LGN. J. Comp. Neurol. 498, 330–351 (2006)
20. Nakamura, H., Gattass, R., Desimone, R., Ungerleider, L.G.: The modular organization of projections from areas V1 and V2 to areas V4 and TEO in macaques. J. Neurosci. 13, 3681–3691 (1993)
21. Rockland, K.S., Virga, A.: Organization of individual cortical axons projecting from area V1 (area 17) to V2 (area 18) in the macaque monkey. Vis. Neurosci. 4, 11–28 (1990)
22. Rockland, K.S.: Configuration, in serial reconstruction, of individual axons projecting from area V2 to V4 in the macaque monkey. Cereb Cortex 2, 353–374 (1992)
23. Rockland, K.S., Saleem, K.S., Tanaka, K.: Divergent feedback connections from areas V4 and TEO in the macaque. Vis. Neurosci. 11, 579–600 (1994)
24. Schummers, J., Mario, J., Sur, M.: Synaptic integration by V1 neurons depends on location within the orientation map. Neuron 36, 969–978 (2002)

25. Przybyszewski, A.W., Potapov, D.O., Rockland, K.S.: Feedback connections from area V4 to LGN. In: Ann. Meet. Society for Neuroscience, San Diego, USA (2001), http://sfn.scholarone.com/itin2001/prog#620.9
26. Lades, M., Vortbrueggen, J.C., Buhmann, J., Lange, J., von der Malsburg, C., Wuertz, R.P., Konen, W.: Distortion invariant object recognition in the dynamic link architecture. IEEE Transactions on Computers 42, 300–311 (1993)
27. Grdenfors, P.: Conceptual Spaces. MIT Press, Cambridge (2000)
28. Polkowski, L., Skowron, A.: Rough Mereological Calculi of Granules: A Rough Set Approach to Computation. Computational Intelligence 17, 472–492 (2001)
29. Przybyszewski, A.W., Linsay, P.S., Gaudiano, P., Wilson, C.: Basic Difference Between Brain and Computer: Integration of Asynchronous Processes Implemented as Hardware Model of the Retina. IEEE Trans. Neural Networks 18, 70–85 (2007)
30. Treisman, A.: Features and objects: the fourteenth Bartlett memorial lecture. Q J. Exp. Psychol. A 40, 201–237 (1988)
31. Ramachandran, V.S.: Perception of shape from shading. Nature 331, 163–166 (1988)
32. Ullman, S., Vidal-Naquet, M., Sali, E.: Visual features of intermediate complexity and their use in classification. Nature Neuroscience 5, 682–687 (2002)
33. Thorpe, S., Faze, D., Merlot, C.: Speed of processing in the human visual system. Nature 381, 520–522 (1996)
34. Rousselet, G.A., Fabre-Thorpe, M., Thorpe, S.J.: Parallel processing in high-level categorization of natural images. Nat. Neurosci. 5, 629–630 (2002)
35. David, S.V., Hayden, B.Y., Gallant, J.L.: Spectral receptive field properties explain shape selectivity in area V4. J. Neurophysiol. 96, 3492–3505 (2006)
36. Merigan, W.H.: Cortical area V4 is critical for certain texture discriminations, but this effect is not dependent on attention. Vis. Neurosci. 17(6), 949–958 (2000)
37. Merigan, W.H., Pham, H.A.: V4 lesions in macaques affect both single- and multiple-viewpoint shape discriminations. Vis. Neurosci. 15(2), 359–367 (1998)
38. Girard, P., Lomber, S.G., Bullier, J.: Shape discrimination deficits during reversible deactivation of area V4 in the macaque monkey. Cereb Cortex 12(11), 1146–1156 (2002)
39. Dumoulin, S.O., Hess, R.F.: Cortical specialization for concentric shape processing Vision Research, vol. 47, pp. 1608–1613 (2007)
40. Biederman, I., Subramaniam, S., Bar, M., Kalocsai, P., Fiser, J.: Subordinate-level object classification reexamined. Psychol. Res. 62, 131–153 (1999)
41. Poggio, T., Edelman, S.: A network that learns to recognize three-dimensional objects. Nature 343, 263–266 (1990)
42. Girard, P., Hup, J.M., Bullier, J.: Feedforward and feedback connections between areas V1 and V2 of the monkey have similar rapid conduction velocities. J. Neurophysiol. 85(3), 1328–1331 (2001)
43. Wiscott, L., Fellous, J.-M., Krueger, N., von der Malsburg, C.: Face recognition by elastic graph matching. IEEE Pattern Recognition and Machine Intelligence 19, 775–779 (1997)

Diagnostic Feature Analysis of a Dobutamine Stress Echocardiography Dataset Using Rough Sets

Kenneth Revett

University of Westminster, Harrow School of Computer Science
London, England, HA1 3TP

Abstract. Stress echocardiography is an important functional diagnostic and prognostic tool that is now routinely applied to evaluate the risk of cardiovascular artery disease (CAD). In patients who are unable to safely undergo a stress based test, dobutamine is administered which provides a similar effect to stress on the cardiovascular system. In this work, a complete dataset containing data on 558 subjects undergoing a prospective longitudinal study is employed to investigate what diagnostic features correlate with the final outcome. The dataset was examined using rough sets, which produced a series of decision rules that predicts which features influence the outcomes measured clinically and recorded in the dataset. The results indicate that the ECG attribute was the most informative diagnostic feature. In addition, prehistory information has a significant impact on the classification accuracy.

Keywords: dobutamine, ECG, LTF-C, Reducts, rough sets, Stress echocardiography.

1 Introduction

Heart disease remains the number one cause of mortality in the western world. Coronary arterial disease (CAD) is a primary cause of morbidity and mortality in patients with heart disease. The early detection of CAD was in part made possible in the late 1970's by the introduction of echocardiography - a technique for measuring the physical properties of the heart using a variety of imaging techniques such as ultrasound, and doppler flow measurements [1], [2], [3]. The purpose of these imaging studies is to identify structural malformations such as aneurysms and valvular deformities. Although useful, structural information may not provide the full clinical picture in the way that functional imaging techniques such as stress echocardiography (SE) may. This imaging technique is a versatile tool that allows clinicians to diagnose patients with CAD efficiently and accurately. In addition, it provides information concerning the prognosis of the patient - which can be used to provide on-going clinical support to help reduce morbidity. The underlying basis for SE is the induction of cardiovascular stress, which generates cardiac ischemia, resulting in cardiac wall motion (a distension type of motion). The motion should reflect the ability of the vasculature to adapt to stressful situations such as enhanced physical activity. The extent to which the vessels (and the heart itself expands) under strenuous activity reflects the viability of the vasculature system. In coronary artery disease, the ability of the vasculature to adapt is limited as a result of episodes of ischemia - reduction in local blood supply - which causes tissue damage. Normally, the walls of the heart (in particular the left ventrical) change (move) in a typical fashion in response to stress (i.e. heavy exercise). A quantitative measure called the wall motion score is computed and its magnitude is directly related to the extent of the WMA score. The WMA provides a quantitative measure of how the heart responds to stress. Stress echocardiography (SE) was originally induced under conditions of strenuous exercise such as bike and treadmills. In many cases though, patients are not able to exercise to the level required and pharmacological agents such as dobutamine or dipyridamole have been used to induce approximately the same level of stress on the heart as physical exercise. Dobutamine in particular emulates physical exercise effects on the cardiovascular system by increasing the heart rate and blood pressure and impacts cardiac contractility - which drives cardiac oxygen demand [4]. A number of reports have indicated that though there are subtle differences between exercise and pharmacologically induced stress, they essentially provide the same stimulus to the heart and can therefore, in general, be used interchangeably [5],[6]. The focus of this paper was to investigate the effectiveness of dobutamine stress echocardiography (DSE) by analysing the results of a large study of 558 patients undergoing DSE. The purpose is to determine which attributes collected in this study correlate most closely with the decision outcome. After a careful investigation of this dataset, a set of rules is presented that relates conditional features (attributes) to decision outcomes. This rule set is generated through the application of rough sets, a data mining technique developed by the late Professor Pawlak [7]. The antecedents of the rule set contains information about which features are involved in the decision outcome. In addition, values of the relevant features provides quantitative information regarding the values that are relevant for each feature for the respective decision class. This provides very useful information regarding the features that are directly relevant in predicting the outcome: in this case whether the principle outcome is whether SE provides prognostic value in lieu of other relevant and routinely collected medical information with respect to the likelihood of cardiac events. In the next section, a literature review of previous work involving the clinical application of stress echocardiography is presented.

1.1 Previous Work

In 1998, Chuah and colleagues published a report on the investigation of a follow-up study of 860 patients who underwent dobutamine stress echocardiography over a 2-year period [8]. The prinicpal features examined in this study were wall motion abnormalities (WMA), cardiovascular risk factors, and clinical status (collected at the time the dobutamine stress test was administered). Any prior myocardial infarctions were determined by patient history or the presence of significant Q waves. The patient group (consisting of 479 men and 381 women, mean age 70 +/- 10, was monitored for a period of 52 months subsequent to the SE test. The follow up resutls indicates that 86 patients had cardiac events, including 36 myocardial infarctions and cardiac death in 50 patients. Those patients with events tended to have a lower rest ejection fraction and more extensive WMAs at rest and with stress. The authors also examined how outcomes (as measured by the likelihood of an event) correlated with respect to the SE results. Of the patients with normal SE results, 4% (12 of 302) had an event. Patients with new or worsening WMAs (321 patients), 44 (14%) had subsequent cardiac events during the follow up period. Lastly, those patients (237) with fixed WMAs (during rest and at stress), 30 (13%) had cardiac events during the follow-up period. The authors then examined the relationship between the feature space and the likelihood of a follow-up event (identifying univariate predictors of cardiac events). The independent predictors were: a history of congestive heart failure, percentage of abnormal segments at peak-stress (measured via SE), and an abnormal left ventricular end-systolic volume response to stress. In the study by Krivokapich and colleagues [3], the pronostic value of dobutamine SE was directly assessed with respect to predicting cardiac events in patients with or suspected of having coronary arterial disease. The study was a retrospective examination of 1,183 patients that underwent DSE (dobutamine stress echocardiography). The patients were monitored for 12 months after a DSE examination in order to determine whether the results of the DSE were predictive (or at least correlated with)of subsequent cardiac events. The authors examined several features using bivariate logistic regression and forward and backward stepwise multiple logistic regression. The independent variables examined were: history of hypertension, diabetes mellitus, myocardial infarction, coronary artery bypass grafting surgery, age, gender, peak dose of dobutamine, rest and peak dobutamine heart rate, blood pressure, rate pressure product, presence of chest pain, abnormal electrocardiogram (ECG), WMA abnormality, and a positive SE. The results from this study indicate that a postive SE and an abnormal ECG were most indicative of a subsequent cardiac event (defined as a myocardial infarction, death or CABG).Patients that had a positive SE and an abnormal ECG had a 42% cardiac incidence rate, versus a 7% cardiac incidence rate for negative SE and ECG. A positive Se alone yielded a 34% cardiac incidence rate during the 12 month follow up period. These results indicate the predictive power of a positive SE in terms of predicting cardiac events within a relatively short time window. In a study by Marwick and colleagues [6] sought to determine whether dobutamine echocardiography could be used as an independent predictor of cardiac mortaility in a group of 3,156 patients (1,801 men and 1,355 women, mean age 63 +/- 12 years) in a nine-year longitudinal follow-up study (1988-1994). At the time of the SE examination, several clinical variables and patient history were recorded for subsequent uni and multi-variate analysis of predictors of cardiac death. During the follow-up period, 259 (8%) deaths attributed to cardiac failure occurred. The authors analysed the patient data with respect to clinical features in order to examine their predictive capacity generally - and to determine if SE was correlated in anyway with the outcome. Age, gender, heart failure therapy were predictive of cardiac failure during the follow-up period. the addition of resting left ventricular fucntion, and SE testing data further improved the predictive capacity of a sequential model (Kaplan-Meier survival curves and Cox proportional hazards models). In those patients with a negative dobutamine echocardiogram (1,581 pateints), the average rate of cardiac mortality was 1% per year, compared to 8% in those patients with SE abnormalities. The final result from this study indicates that the inclusion of SE, in addition to standard clinical data increaes signficiantly the predictive outcome of cardiac events. Though not an exhaustive list of published examinations of the predictive capacity of dobutamine echocardiography, the cases presented here are indicative of the approach used to examine whether this technique provides positve predictive information that can assist clinicians in patient care (see [8], [9] for additional studies]). The approach is typically a longitudinal study, utilising a substantial patient cohort. As most subjects are in clinical care for suspected heart disease, there is a substantial amount of clinical information that is acquired as part of the routine care these patients. Typically, clinical data provides a predictive capacity on the order of 60%. The deployment of stress echocardiography enhances the predictive capacity over typical clinical data - even that acquired within the context of the disease based on previous medical exposure. Univariate and multivariate models provide quantitative information with respect to which variables appear to be correlated with the decision outcome. The reality for busy clinicians is that they may not be prepared to perform the complex analyses required to extract useful information from their data. This study attempts to provide a rational basis for the examination of the feature space of a typical SE dataset. The goal is to determine if the features are indeed correlated with the decision outcomes - and if so - what subset of features are relevant and what range of values are expected for predictive features. The next section presents a description of the dataset and some of the pre-processing stages employed for subsequent data analysis.

1.2 The Dataset

The data employed in this study was obtained from a prospective dobutamine stress echocardiography (DSE) study at the UCLA Adult Cardiac Imaging and Hemodynamics Laboratory held between 1991 and 1996. The patients were monitored during a five year period and then observed for a further twelve months to determine if the DSE results could predict patient outcome. The outcomes were categorised into the following cardiac events: cardiac death, myocardial infarction (MI), and revascularisation by percutaneous transluminal coronary angioplasty (PTCA) or coronary artery bypass graft surgery (CABG) [5]. After normal exclusionary processes, the patient cohort consisted of 558 subjects (220 women and 338 men) with a median age of 67 (range 26-93). Dobutamine was administered intraveneously using a standard delivery system yielding a maximum dose of 40 g/kg/min. There were a total of 30 attributes collected in this study which are listed in Table 1.

Table 1. The decision table attributes and their data types (continuous, ordinal, or discrete) employed in this study (see for details). Note the range of correlation coefficients was -0.013 to 0.2476 (specific data not shown).

Attribute name	Attribute type
bhr basal heart rate	Integer
basebp basal blood pressure	Integer
basedp basal double product (= bhr x basebp)	Integer
pkhr peak heart rate	Integer
sbp systolic blood pressure	Integer
dp double product (= pkhr x sbp)	Integer
dose dose of dobutamine given	Integer
maxhr maximum heart rate	Integer
mphr(b) % of maximum predicted heart rate	Integer
mbp maximum blood pressure	Integer
dpmaxdo double product on maximum dobutamine dose	Integer
dobdose dobutamine dose at which maximum double product	Integer
age	Integer
gender (male = 0)	Level (2)
baseef baseline cardiac ejection fraction	Integer
dobef ejection fraction on dobutamine	Integer
chestpain (0 experienced chest pain)	Integer
posecg signs of heart attack on ecg (0 = yes)	Integer
equivecg ecg is equivocal (0 = yes)	Integer
restwma wall motion anamoly on echocardiogram (0 = yes)	Integer
posse stress echocardiogram was positive (0 = yes)	Integer
newMI new myocardial infarction, or heart attack (0 = yes)	Integer
newPTCA recent angioplasty (0 = yes)	Level (2)
newCABG recent bypass surgery (0 = yes)	Level (2)
death died (0 = yes)	Level (2)
hxofht history of hypertension (0 = yes)	Level (2)
hxofptca history of angioplasty (0 = yes)	Level (2)
hxofcabg history of bypass surgery (0 = yes)	Level (2)
hxofdm history of diabetes (0 = yes)	Level (2)
hxofMI history of heart attack (0 = yes)	Level (2)

The attributes were a mixture of categorical and continuous values. The decision class used to evaluate this dataset was the outcomes as listed as listed above and in Table 1. As a preliminary evaluation of the dataset, the data was evaluated with respect to each of the four possible measured outcomes included in the decision table individually, excluding each of the other three possible outcomes. This process was repeated for each of the outcomes in the decision table. Next, the effect of the echocardiogram (ECG) was investigated. Reports indicate that this is a very informative attribute with respect to predicting the clinical outcome of a patient [3]. To evaluate the effect of ECG on the outcomes, the base case investigation (all four possible outcomes) was investigated with (base case) and without the ECG attribute. Lastly, the information content of any prehistory information was investigated to examined if there was a correlation between the DSE and the outcome. There were a total of six different history attributes (see Table 1) that were tested to determine if each in isolation had a positive correlation with the outcomes. In the next section, we describe the experiments that were performed using rough sets (RSES 2.2.1).

2 Results

In the first experiment, each outcome was used as the sole decision attribute. The four outcomes were: new Myocardial Infarction (MI) (28 cases), death (24 cases), newPTCA (27 cases), and newCABG (33 cases). All continuous attributes were discretised using the MDL algorithm within RSES [9], [10]. Note there were no missing values in the dataset. A 10-fold cross validation was performed - using decision rules and dynamic reducts. Without any filtering of the reducts or rules, Table 2 presents randomly selected confusion matrices that were generated for each of the decision outcomes for the base case.

The number of rules was quite large - and initially no filtering was performed to reduce either the number of reducts nor the number of rules. The number of reducts for panels 'A' - 'D' in Table 2 were: 104, 159, 245, and 122 respectively. On average, the length of the reducts ranged from 5-9, out of a total of 27 attributes (minus the 3 other outcome decision classes). The number of rules (all of which were deterministic) was quite large, with a range of 23,356-45,330 for the cases listed in table 2. Filtering was performed on both reducts (based on support) and rule coverage in order to reduce the cardinality of the decision rules. The resulting decision rule set were reduced to a range of 314-1,197. The corresponding accuracy was reduced by approximately 4% (range 3- 6%). Filtering can be performed on a variety of conditions, such as LHS support, coverage, RHS support. For a discussion of rule filtering, please consult [10], [11] for an excellent discussion of this topic.

The number of rules was quite large - and initially no filtering was performed to reduce either the number of reducts nor the number of rules. The number of

Table 2. Confusion matrices for the 'base' cases of the four different outcomes. The label 'A' corresponds to death, 'B' to MI, 'C' to new PTCA, and 'D' to newCABG. Note the overall accuracy is placed at the lower right hand corner of each subtable (italicized).

A	0	1			B	0	1
0	204	7	0.97	0	205	4	0.98
1	2	10	0.80	1	0	14	1.0
	0.95	0.22	*0.92*		0.94	0	*0.92*
C	0	1			D	0	1
0	207	9	0.96	0	191	25	0.88
1	6	1	0.14	1	7	0	0
	0.97	0.10	*0.93*		0.96	0.0	*0.86*

Table 3. The classification accuracy obtained from the classification using the exact same protocol for the table reported in Table 2 (note the ECG attribute was included in the decision table). The results are the average over the four different outcomes.

A	0	1			B	0	1
0	206	5	0.98	0	205	4	0.98
1	3	9	0.75	1	0	14	1.0
	0.95	0.22	*0.92*		0.94	0	*0.96*
C	0	1			D	0	1
0	209	7	0.98	0	191	25	0.88
1	1	6	0.86	1	0	7	1.00
	0.97	0.10	*0.93*		0.96	0.0	*0.94*

Table 4. The classification accuracy obtained from the classification using the same protocol for the data reported in table 2 (note the ECG attribute was included in the decision table). The results are the average over the four different outcomes.

Attribute name	Classification accuracy
History of hypertension	91.1%
History of diabetes	85.3%
History of smoking	86.3%
History of angioplasty	90.3%
History of coronary artery bypass surgery	82.7%

reducts for panels 'A' - 'D' in Table 2 were: 104, 159, 245, and 122 respectively. On average, the length of the reducts ranged from 5-9, out of a total of 27 attributes (minus the 3 other outcome decision classes). The number of rules (all of which were deterministic) was quite large, with a range of 23,356-45,330 for the cases listed in table 2. Filtering was performed on both reducts (based on support) and rule coverage in order to reduce the cardinality of the decision rules. The resulting decision rule set were reduced to a range of 314-1,197. The corresponding accuracy was reduced by approximately 4% (range 3-6%). In the next experiment, the correlation between the outcome and the ECG result was examined. It has been reported that the ECG, which is a standard cardiological test to measure functional activity of the heart, should be correlated with the outcome [2]. We therefore repeated the experiment in Table 2, with the ECG attribute excluded (masked) from the decision table. The results are reported in Table 3. Lastly, we examined the effect of historical information that was collected and incorporated into the dataset (see Table 1). These historical attributes include: history of hypertension, diabetes, smoking, myocardial infarction, angioplasty, and coronary artery bypass surgery. We repeated the base set of experiments (including ECG) and withheld each of the historical attributes one at a time and report the results as a set of classification accuracies, listed in Table 4.

In addition to classification accuracy, rough sets provide a collection of decision rules in conjunctive normal form. These rules contain the attributes and

Table 5. Sample set of rules from the base case (+ECG) with death as the decision outcome. The right hand column indicates the support (LHS) for the corresponding rule. Note that these rules were selected randomly from the full set.

Rule	*Support*
dp([20716, *]) AND dobdoes(40) AND hxofDM(0) AND anyevent(0) ⇒ death(0)	19
dp([*,13105]) AND dobdoes(40) AND hxofDM(0) AND anyevent(0)⇒ death(0)	18
basebp([*,159]) AND sbp([115,161]) AND hxofDM(0) AND anyevent(0) ⇒ death(0)	24
dp([*,131105) AND dobdose(35) AND dobEF([53,61]) AND hxofDM(1)⇒ death(10)	10
dp([20633,20716]) AND dobdoes(4) AND baseEF([56,76]) AND hxofDM(0) AND anyevent(1) ⇒ death (1)	1
dp([*,13]) AND dobdoes(30) AND hxofCABG(0) AND anyevent(1) AND ecg([*,2]) ⇒ death(1)	12

their values that are antecedents in a rule base. Therefore, the decision rules provide a codification of the knowledge contained within the decision table. Examples of the resulting rule set for the base case, using MI as the decision attribute is presented in table 5.

3 Conclusion

This dataset contained a complete set of attributes (30) that was a mixture of continuous and categorical data. The data was obtained from a prospective study of cardiovascular health obtained by professional medical personal (cardiographers). The attributes were obtained from patients undergoing stress echocardiography, a routine medical technique employed to diagnose cardiovascular artery disease. From the initial classification results, the specificity of the classification using rough sets was quite high (90+%) - consistent with some literature reports [2],[6]. As can be seen in Table 2, the sensitivity of the test was reasonably high, and consistent with several literature reports. The effect of ECG, the attribute most correlated with the clinical outcome of CAD, was measured by masking this attribute. The results indicate that this attribute did not have a significant impact on the overall classification accuracy, but did manage to increase the sensitivity was reduced slightly when it was excluded in the decision table. This result requires further examination to quantify the role of an abnormal ECG - and the interaction/information content of an abnormal ECG and other medical indicators.The effect of patient history was examined, and the results (see Table 4) indicate that in general, relevant medical history did have a positive impact on the classification accuracy. This result was quantified by examining the classification accuracy when these 5 history factors were removed from the decision table (one at a time). The effect of their combination was not examined in this paper, which is left for future work. The data clearly indicate that a positive SE result was highly correlated with a subsequent cardiac event. This result when demonstrated by examining the rule set, looking at the occurrences of this attribute in the consequent. Lastly, the rule set that was produced yielded a consistently reduced set of attributes - ranging from 4-9 attributes, greatly reducing the size of the dataset. As displayed in Table 5 - and generally across the rule set, the dp and dobdose attributes appear consistently (has a large support) within all decision outcomes (data not displayed). This type of analysis is a major product of the rough sets approach to data analysis - extraction of knowledge from data. This is a preliminary study that will be pursued in conjunction with a qualified cardiologist. The results generated so far are interesting - and certainly consistent and in many cases superior to other studies [1],[3]. To this author's knowledge, this is the first report which examined the dobutamine SE literature using rough sets. Komorowski & Ohn have examined a similar dataset - but the imaging technique and attributes selected were different from those used in the study investigated in this work [12]. In a preliminary examination of this dataset, Revett [13] published similar results to this study. A principal addition in this study is confirmation of the 2007 study through the application of a novel neural network (LTF-C) to corroborate the reduced attribute set extracted from the rough sets examination. The application of LTF-C did indeed confirm that the classification accuracy was maximal with the selected set of attributes, compared to an exhaustive investigation of the other attributes with respect to training speed and classification accuracy. The results from this study indicate that a rough sets approach to rule extraction from this dataset provided evidence that corroborate much of the results reported in the literature. The basis for applying rough sets is that it provides evidence with regards to the features and their values that are predictive with respect to the decision class. Further analysis of this dataset is possible, and this analysis would benefit from a close collaboration between medical experts and data mining engineers.

Acknowledgements. The author would like to acknowledge the source of the dataset used in this study: Alan Garfinkle, UCLA (at the time of submission): http://www.stat.ucla.edu:16080/projects/datasets/cardiacexplanation.html

References

1. Tsutsui, J.M., Elhendy, A., Anderson, J.A., Xie, F., McGrain, A.C., Porter, T.R.: Prognostic value of dobutamine stress myocardial contrast perfuson echocardiography. Circulation 112, 1444–1450 (2005)
2. Armstrong, W.F., Zoghbi: Stress Echocardiography: Current Methodology and Clinical Applications. J. Am. Coll. Cardiology 45, 1739–1747 (2005)
3. Krivokapich, J., Child, J.S., Walter, D.O., Garfinkel, A.: Prognostic value of dobutamine stress echocardiography in predicting cardiac events in patients with known or suspected coronary artery disease. J. Am. Coll. Cardiology 33, 708–716 (1999)

4. Bergeron, S., Hillis, G., Haugen, E., Oh, J., Bailey, K., Pellikka, P.: Prognostic value of dobutamine stress echocardiography in patients with chronic kidney disease. American Heart Journal 153(3), 385–391 (1982); Pawlak, Z.: Rough Sets. International Journal of Computer and Information Sciences 11, 341–356 (1982)
5. Marwick, T.H., Case, C., Poldermans, D., Boersma, E., Bax, J., Sawada, S., Thomas, J.D.: A clinical and echocardiographic score for assigning risk of major events after dobutamine echocardiography. Journal of the American College of cardiology 43(11), 2102–2107 (2004)
6. Marwick, T.H., Case, C., Sawada, S., Timmerman, C., Brenneman, P., Kovacs, R., Short, L., Lauer, M.: Prediction of mortaility using dobutamine echocardiography. Journal of the American College of Cardiology 37(3), 754–760 (2001)
7. Pawlak, Z.: Rough sets - Theoretical aspects of reasoning about data. Kluwer, Dordrecht (1991)
8. Chuah, S.-C., Pellikka, P.A., Roger, V.L., McCully, R.B., Seward, J.B.: Role of dobutamine stress echocardiography in rpedicting utcome of 860 patients with known or suspsected coronary artery disease. In: Circulation 1997, pp. 1474–1480 (1998)
9. Senior, R.: Stress echocardiography - current status. Business Briefing: European Cardiology, 26–29 (2005)
10. Bazan, J., Szczuka, M.: The Rough Set Exploration System. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets III. LNCS, vol. 3400, pp. 37–56. Springer, Heidelberg (2005), `http://logic.mimuw.edu.pl/~rses`
11. Komorowski, J., Pawlak, Z., Polkowski, L., Skowron, A.: Rough sets: A tutorial. In: Pal, S.K., Skowron, A. (eds.) Rough Fuzzy Hybridization - A New Trend in Decision Making, pp. 3–98. Springer, Heidelberg (1999)
12. Komorowski, J., Øhrn, A.: Modelling prognostic power of cardiac tests using rough sets. Artificial Intelligence in Medicine 15, 167–191 (1999)
13. Revett, K.: Analysis of a dobutamine stress echocardiography dataset using rough sets. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 756–762. Springer, Heidelberg (2007)

Rules and Apriori Algorithm in Non-deterministic Information Systems

Hiroshi Sakai[1], Ryuji Ishibashi[1], Kazuhiro Koba[1], and Michinori Nakata[2]

[1] Mathematical Sciences Section, Department of Basic Sciences, Faculty of Engineering, Kyushu Institute of Technology, Tobata, Kitakyushu 804, Japan
sakai@mns.kyutech.ac.jp
[2] Faculty of Management and Information Science, Josai International University, Gumyo, Togane, Chiba 283, Japan
nakatam@ieee.org

Abstract. This paper presents a framework of rule generation in *Non-deterministic Information Systems* (*NISs*), which follows rough sets based rule generation in *Deterministic Information Systems* (*DISs*). Our previous work about *NISs* coped with *certain rules*, *minimal certain rules* and *possible rules*. These rules are characterized by the concept of *consistency*. This paper relates possible rules to *rules* by the criteria *support* and *accuracy* in *NISs*. On the basis of the information incompleteness in *NISs*, it is possible to define new criteria, i.e., *minimum support*, *maximum support*, *minimum accuracy* and *maximum accuracy*. Then, two strategies of rule generation are proposed based on these criteria. The first strategy is *Lower Approximation strategy*, which defines rule generation under the worst condition. The second strategy is *Upper Approximation strategy*, which defines rule generation under the best condition. To implement these strategies, we extend *Apriori* algorithm in *DISs* to *Apriori* algorithm in *NISs*. A prototype system is implemented, and this system is applied to some data sets with incomplete information.

Keywords: Rough sets, Non-deterministic information, Incomplete information, Rule generation, Lower and upper approximations, Apriori algorithm.

1 Introduction

Rough set theory has been used as a mathematical tool of soft computing for approximate two decades. This theory usually handles tables with deterministic information. Many applications of this theory, such as rule generation, machine learning and knowledge discovery, have been presented [5, 9, 15, 21, 22, 23, 24, 25, 36, 38].

We follow rule generation in *Deterministic Information Systems* (*DISs*) [21, 22, 23, 24, 33], and we describe rule generation in *Non-deterministic Information*

Systems ($NISs$). $NISs$ were proposed by Pawlak [21], Orłowska [19, 20] and Lipski [13, 14] to handle information incompleteness in $DISs$, like null values, unknown values, missing values. Since the emergence of incomplete information research, $NISs$ have been playing an important role. Therefore, rule generation in $NISs$ will also be an important framework for rule generation from incomplete information.

The following shows some important researches on rule generation from incomplete information. In [13, 14], Lipski showed a question-answering system besides an axiomatization of logic, and Orłowska established rough set analysis for non-deterministic information [3, 19, 20]. Grzymala-Busse developed a system named $LERS$ which depends upon $LEM1$ and $LEM2$ algorithms [5, 6, 7], and recently proposed four interpretations of missing attribute values [8]. Stefanowski and Tsoukias defined non symmetric similarity relations and valued tolerance relations for analyzing incomplete information [34, 35]. Kryszkiewicz proposed a framework of rules in incomplete information systems [10, 11, 12]. According to authors' knowledge, these are the most important researches on incomplete information. We have also discussed several issues related to non-deterministic information and incomplete information [16, 17, 18], and proposed a framework named *Rough Non-deterministic Information Analysis* ($RNIA$) [26, 27, 28, 29, 30, 31, 32].

In this paper, we briefly review $RNIA$ including certain and possible rules, then develop rule generation by the criteria *support* and *accuracy* in $NISs$. In this rule generation, we extend *Apriori* algorithm in $DISs$ to a new algorithm in $NISs$. The computational complexity of this new algorithm is almost the same as *Apriori* algorithm. Finally, we investigate a prototype system, and apply it to some data sets with incomplete information.

2 Basic Definitions and Background of the Research

This section summarizes basic definitions, and reviews the background of this research in [28, 31, 32].

2.1 Basic Definitions

A *Deterministic Information System* (DIS) is a quadruplet $(OB, AT, \{VAL_A | A \in AT\}, f)$, where OB is a finite set whose elements are called *objects*, AT is a finite set whose elements are called *attributes*, VAL_A is a finite set whose elements are called *attribute values* and f is such a mapping that $f : OB \times AT \rightarrow \cup_{A \in AT} VAL_A$ which is called a *classification function*. If $f(x, A){=}f(y, A)$ for every $A \in ATR \subset AT$, we see there is a relation between x and y for ATR. This relation is an equivalence relation over OB, and this is called an *indiscernibility relation*.

We usually define two sets $CON \subseteq AT$ which we call *condition attributes* and $DEC \subseteq AT$ which we call *decision attributes*. An object $x \in OB$ is *consistent* (with any distinct object $y \in OB$), if $f(x, A){=}f(y, A)$ for every $A \in CON$ implies $f(x, A){=}f(y, A)$ for every $A \in DEC$.

A *Non-deterministic Information System* (NIS) is also a quadruplet (OB, $AT, \{VAL_A | A \in AT\}, g$), where $g : OB \times AT \rightarrow P(\cup_{A \in AT} VAL_A)$ (a power set of $\cup_{A \in AT} VAL_A$). Every set $g(x, A)$ is interpreted as that there is an actual value in this set, but this value is not known. For a NIS=$(OB, AT, \{VAL_A | A \in AT\}, g)$ and a set $ATR \subseteq AT$, we name a DIS=$(OB, ATR, \{VAL_A | A \in ATR\}, h)$ satisfying $h(x, A) \in g(x, A)$ a *derived DIS (for ATR) from NIS*. For a set ATR=$\{A_1, \cdots, A_n\} \subseteq AT$ and any $x \in OB$, let $PT(x, ATR)$ denote the Cartesian product $g(x, A_1) \times \cdots \times g(x, A_n)$. We name every element a *possible tuple* (*for ATR*) *of x*. For a possible tuple ζ=$(\zeta_1, \cdots, \zeta_n) \in PT(x, ATR)$, let $[ATR, \zeta]$ denote a formula $\bigwedge_{1 \leq i \leq n} [A_i, \zeta_i]$. Every $[A_i, \zeta_i]$ is called a *descriptor*. Let $PI(x, CON, DEC)$ ($x \in OB$) denote a set $\{[CON, \zeta] \Rightarrow [DEC, \eta] | \zeta \in PT(x, CON),\ \eta \in PT(x, DEC)\}$. We name an element of $PI(x, CON, DEC)$ a *possible implication* (*from CON to DEC*) *of x*. In the following, τ denotes a possible implication, and τ^x denotes a possible implication obtained from an object x.

Now, we define six classes of possible implications, certain rules and possible rules. For any $\tau^x \in PI(x, CON, DEC)$, let $DD(\tau^x, x, CON, DEC)$ denote a set $\{\varphi | \ \varphi$ is such a derived DIS for $CON \cup DEC$ that an implication from x in φ is equal to $\tau^x\}$. If $PI(x, CON, DEC)$ is a singleton set $\{\tau^x\}$, we say τ^x is *definite*. Otherwise we say τ^x is *indefinite*. If a set $\{\varphi \in DD(\tau^x, x, CON, DEC) | \ x$ is consistent in $\varphi\}$ is equal to $DD(\tau^x, x, CON, DEC)$, we say τ^x is *globally consistent* (GC). If this set is equal to $\{\}$, we say τ^x is *globally inconsistent* (GI). Otherwise, we say τ^x is *marginal* (MA). By combining two cases, i.e., 'D(*efinite*) or I(*ndefinite*)' and 'GC, MA or GI', we define six classes, DGC, DMA, DGI, IGC, IMA, IGI in Table 1. A possible implication τ^x belonging to DGC class is consistent in all derived DISs, and this τ^x is not influenced by the information incompleteness, therefore we name τ^x a *certain rule* or more correctly a *candidate of* a *certain rule*. A possible implication τ^x belonging to either DGC, IGC, DMA or IMA class is consistent in some $\varphi \in DD(\tau^x, x, CON, DEC)$. Therefore, we name τ^x a *possible rule* or more correctly a *candidate of* a *possible rule*.

Table 1. Six classes of possible implications in NISs

	GC	*MA*	*GI*
Definite	*DGC*	*DMA*	*DGI*
Indefinite	*IGC*	*IMA*	*IGI*

Now, we give necessary and sufficient conditions for characterizing GC, MA and GI classes. For any $\zeta \in PT(x, ATR)$, we define two sets

$$inf(x, ATR, \zeta)=\{y \in OB | PT(y, ATR)=\{\zeta\}\} \cup \{x\},$$
$$sup(x, ATR, \zeta)=\{y \in OB | \zeta \in PT(y, ATR)\}.$$

Intuitively, $inf(x, ATR, \zeta)$ implies a set of objects whose tuples are ζ and definite. If a tuple $\zeta \in PT(x, ATR)$ is not definite, this object x does not satisfy

$PT(x, ATR)$=$\{\zeta\}$. Therefore, we added a set $\{x\}$ in the definition of inf. A set $sup(x, ATR, \zeta)$ implies a set of objects whose tuples may be ζ. Even though x does not appear in the right hand side of sup, we employ the $sup(x, ATR, \zeta)$ notation due to the $inf(x, ATR, \zeta)$ notation. Generally, $\{x\} \subseteq inf(x, ATR, \zeta)$= $sup(x, ATR, \zeta)$ holds in $DISs$, and $\{x\} \subseteq inf(x, ATR, \zeta) \subseteq sup(x, ATR, \zeta)$ holds in $NISs$.

Theorem 1 [28, 29]. For a NIS, let us consider a possible implication τ^x:$[CON, \zeta] \Rightarrow [DEC, \eta] \in PI(x, CON, DEC)$. Then, the following holds.

(1) τ^x belongs to GC class, if and only if $sup(x, CON, \zeta) \subseteq inf(x, DEC, \eta)$.
(2) τ^x belongs to MA class, if and only if $inf(x, CON, \zeta) \subseteq sup(x, DEC, \eta)$.
(3) τ^x belongs to GI class, if and only if $inf(x, CON, \zeta) \not\subseteq sup(x, DEC, \eta)$.

Proposition 2 [28, 29]. For any NIS, let $ATR \subseteq AT$ be $\{A_1, \cdots, A_n\}$, and let a possible tuple $\zeta \in PT(x, ATR)$ be $(\zeta_1, \cdots, \zeta_n)$. Then, the following holds.

(1) $inf(x, ATR, \zeta)$=$\cap_i inf(x, \{A_i\}, (\zeta_i))$.
(2) $sup(x, ATR, \zeta)$=$\cap_i sup(x, \{A_i\}, (\zeta_i))$.

2.2 An Illustrative Example

Let us consider NIS_1 in Table 2. There are four derived $DISs$ in Table 3.

Table 2. A table of NIS_1

OB	$Color$	$Size$
1	$\{red, green\}$	$\{small\}$
2	$\{red, blue\}$	$\{big\}$
3	$\{blue\}$	$\{big\}$

Table 3. Four derived $DISs$ from NIS_1. Tables are φ_1, φ_2, φ_3, φ_4 to the right.

OB	$Color$	$Size$	OB	$Color$	$Size$	OB	$Color$	$Size$	OB	$Color$	$Size$
1	red	$small$	1	red	$small$	1	$green$	$small$	1	$green$	$small$
2	red	big	2	$blue$	big	2	red	big	2	$blue$	big
3	$blue$	big	3	$blue$	big	3	$blue$	big	3	$blue$	big

Let us focus on a possible implication

$$\tau_1^3 : [Color, blue] \Rightarrow [Size, big] \in PI(3, \{Color\}, \{Size\}).$$

This τ_1^3 means the first implication from object 3, and τ_1^3 appears in four derived $DISs$. Since the following holds,

$$\{2, 3\} = sup(3, \{Color\}, (blue)) \subseteq inf(3, \{Size\}, (big)) = \{2, 3\},$$

τ_1^3 belongs to DGC class according to Theorem 1. Namely, τ_1^3 is consistent in each derived DIS. As for the second possible implication,

$$\tau_2^1 : [Color, red] \Rightarrow [Size, small] \in PI(1, \{Color\}, \{Size\}),$$

the following holds:

$$\{1,2\} = sup(1, \{Color\}, (red)) \not\subseteq inf(1, \{Size\}, (small)) = \{1\},$$
$$\{1\} = inf(1, \{Color\}, (red)) \subseteq sup(1, \{Size\}, (small)) = \{1\}.$$

According to Theorem 1, τ_2^1 belongs to IMA class, namely τ_2^1 appears in φ_1 and φ_2, and τ_2^1 is consistent just in φ_2.

2.3 Certain Rule Generation in Non-deterministic Information Systems

This subsection briefly reviews the previous research on certain rule generation in $NISs$ [28, 29]. We have named possible implications in DGC class certain rules. For certain rule generation, we dealt with the following problem.

Problem 1 [29]. For a NIS, let DEC be decision attributes and let η be a tuple of decision attributes values for DEC. Then, find minimal certain rules in the form of $[CON, \zeta] \Rightarrow [DEC, \eta]$.

According to Theorem 1, Problem 1 is reduced to find some minimal sets of descriptors $[CON, \zeta]$ satisfying $sup(x, CON, \zeta) \subseteq inf(x, DEC, \eta)$. For solving this problem, we employed a discernibility function in $DISs$ [33]. We adjusted the discernibility function to $NISs$, and implemented utility programs [29].

Example 1. Let us focus on a possible implication $\tau_1^3 : [Color, blue] \Rightarrow [Size, big]$ in Table 2, again. Since $inf(3, \{Size\}, (big))=\{2, 3\}$, it is necessary to discriminate object $1 \notin \{2, 3\}$ from object 3. The descriptor $[Color, blue]$ discriminates object 1 from object 3, because $sup(3, \{Color\}, (blue))=\{2, 3\}$ and $1 \notin sup(3, \{Color\}, (blue))$ hold. In this way, the discernibility function $DF(3)$ becomes $[Color, blue]$, and we obtain minimal certain rule τ_1^3. The following is a real execution.

```
% ./plc
?-consult(dgc_rule.pl).
yes
?-trans.
File Name for Read Open:'data.pl'.
Decision Definition File:'attrib.pl'.
File Name for Write Open:'data.rs'.
EXEC_TIME=0.01796603203(sec)
yes
?-minimal.        /* [1,blue](=[Color,blue]),[2,big](=[Size,big]) */
<<Minimal Certain Rules from object 3>>
 Descriptor [1,blue] is a core for object 1
 [1,blue]=>[2,big] [4/4(=4/4,1/1),Definite,GC: Only Core Descriptors]
EXEC_TIME=0.01397013664(sec)
yes
```

This program is implemented in prolog [28, 29, 30]. Each attribute is identified with its ordinal number, namely *Color* and *Size* are identified with 1 and 2, respectively. The underlined parts are specified by a user.

2.4 Non-deterministic Information and Incomplete Information

This subsection clarifies the semantic difference of non-deterministic information and incomplete information.

Table 4. A table of *DIS* with incomplete information

OB	*Color*	*Size*
1	$*$	*small*
2	$*$	*big*
3	*blue*	*big*

Let us consider Table 4. The symbol "$*$" is often employed for indicating incomplete information. Table 4 is generated by replacing non-deterministic information in Table 2 with $*$. There are some interpretations of this $*$ symbol [4, 7, 8, 10, 17, 34]. In the most simple interpretation of incomplete information, the symbol $*$ may be each attribute value. Namely, $*$ may be either *red*, *blue* or *green*, and there are 9 (=3×3) possible tables in Table 4. In such a possible table, the implication from object 1 may be $[Color, blue] \Rightarrow [Size, small]$, and this contradicts $\tau_1^3 : [Color, blue] \Rightarrow [Size, big]$. On the other hand in Table 2, the function is $g(1, \{Color\})=\{red, green\} \subsetneq \{red, blue, green\}$, and we dealt with four derived *DISs*. In Table 2, we did not handle $[Color, blue] \Rightarrow [Size, small]$ from object 1. Like this, τ_1^3 is globally consistent in Table 2, but τ_1^3 is inconsistent in Table 4.

The function $g(x, A)$ and a set $sup(x, ATR, \zeta)$ are employed for handling information incompleteness, and cause the semantic difference of non-deterministic information and incomplete information. In *RNIA*, the interpretation of the information incompleteness comes from the meaning of the function $g(x, A)$. There is no other assumption on this interpretation.

2.5 A Problem of Possible Rule Generation in Non-deterministic Information Systems

We have defined possible rules by possible implications which belong to either *DGC*, *DMA*, *IGC* or *IMA* classes. In this case, there may be a large number of possible implications satisfying condition (2) in Theorem 1. For example in Table 2, there are four possible implications including τ_1^3 and τ_2^1, and every possible implication is consistent in at least a derived *DIS*. Thus, every possible implication is a possible rule. This implies the definition of possible rules may be too weak. Therefore, we need to employ other criteria for defining rules except certain rules.

In the subsequent sections, we follow the framework of rule generation [1, 2, 22, 36, 38], and employ criteria, *support* and *accuracy* for defining rules including possible rules.

3 New Criteria: Minimum Support, Minimum Accuracy, Maximum Support and Maximum Accuracy

This section proposes new criteria in $NISs$, and investigates the calculation of criteria. These new criteria depend upon each element in $DD(\tau^x, x, CON, DEC)$, but the complexity of the calculation does not depend upon the number of elements in $DD(\tau^x, x, CON, DEC)$.

3.1 Definition of New Criteria

In a DIS, criteria *support* and *accuracy* are usually applied to defining rules [1, 2, 36]. In a NIS, we define the following four criteria, i.e., minimum *support*: $minsupp(\tau^x)$, maximum *support*: $maxsupp(\tau^x)$, minimum *accuracy*: $minacc(\tau^x)$ and maximum *accuracy*: $maxacc(\tau^x)$ in the following:

(1) $minsupp(\tau^x) = Minimum_{\varphi \in DD(\tau^x, x, CON, DEC)}\{support(\tau^x) \text{ in } \varphi\}$,
(2) $maxsupp(\tau^x) = Maximum_{\varphi \in DD(\tau^x, x, CON, DEC)}\{support(\tau^x) \text{ in } \varphi\}$,
(3) $minacc(\tau^x) = Minimum_{\varphi \in DD(\tau^x, x, CON, DEC)}\{accuracy(\tau^x) \text{ in } \varphi\}$,
(4) $maxacc(\tau^x) = Maximum_{\varphi \in DD(\tau^x, x, CON, DEC)}\{accuracy(\tau^x) \text{ in } \varphi\}$.

If τ^x is definite, $DD(\tau^x, x, CON, DEC)$ is equal to all derived $DISs$. If τ^x is indefinite, $DD(\tau^x, x, CON, DEC)$ is a subset of all derived $DISs$. If we employ all derived $DISs$ instead of $DD(\tau^x, x, CON, DEC)$ in the above definition, $minsupp(\tau^x)$ and $minacc(\tau^x)$ are 0, respectively. Because, there exist some derived $DISs$ where τ^x does not appear. This property for each indefinite τ^x is trivial, so we define $minsupp(\tau^x)$ and $minacc(\tau^x)$ over $DD(\tau^x, x, CON, DEC)$.

Example 2. In Table 2, let us focus on a possible implication

$\tau_1^3 : [Color, blue] \Rightarrow [Size, big] \in PI(3, \{Color\}, \{Size\})$.

In $DD(\tau_1^3, 3, \{Color\}, \{Size\})$=$\{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$, the following holds:

$1/3 = minsupp(\tau_1^3) \leq maxsupp(\tau_1^3) = 2/3$,

$1 = minacc(\tau_1^3) \leq maxacc(\tau_1^3) = 1$.

As for the second possible implication,

$\tau_2^1 : [Color, red] \Rightarrow [Size, small] \in PI(1, \{Color\}, \{Size\})$,

in $DD(\tau_2^1, 1, \{Color\}, \{Size\})$=$\{\varphi_1, \varphi_2\}$, the following holds:

$1/3 = minsupp(\tau_2^1) \leq maxsupp(\tau_2^1) = 1/3$,

$1/2$= $minacc(\tau_2^1) \leq maxacc(\tau_2^1) = 1$.

3.2 A Simple Method for Calculating Criteria

In order to obtain *minsupp*(τ^x), *minacc*(τ^x), *maxsupp*(τ^x) and *maxacc*(τ^x), the most simple method is to examine each *support*(τ^x) and *accuracy*(τ^x) in every $\varphi \in DD(\tau^x, x, CON, DEC)$. This method is simple, however the number of elements in $DD(\tau^x, x, CON, DEC)$ is $\prod_{A \in CON, B \in DEC, x \neq y} |g(y, A)||g(y, B)|$, and the number of elements increases in exponential order. Therefore, this simple method will not be applicable to $NISs$ with a large number of derived $DISs$.

3.3 Effective Calculation of Minimum Support and Minimum Accuracy

Let us consider how to calculate *minsupp*(τ^x) and *minacc*(τ^x) for $\tau^x : [CON, \zeta] \Rightarrow [DEC, \eta]$ from object x. Each object y with descriptors $[CON, \zeta]$ or $[DEC, \eta]$ influences *minsupp*(τ^x) and *minacc*(τ^x). Table 5 shows all possible implications with descriptors $[CON, \zeta]$ or $[DEC, \eta]$. For example in $CASE$ 1, we can obtain just an implication. However in $CASE$ 2, we can obtain either (C2.1) or (C2.2). Every possible implication depends upon the selection of a value in $g(y, DEC)$. This selection of attribute values specifies some derived $DISs$ from a NIS.

Table 5. Seven cases of possible implications (related to $[CON, \zeta] \Rightarrow [DEC, \eta]$ from object x, $\eta \neq \eta'$, $\zeta \neq \zeta'$) in $NISs$

	$Condition : CON$	$Decision : DEC$	$Possible\_Implications$
$CASE1$	$g(y, CON) = \{\zeta\}$	$g(y, DEC) = \{\eta\}$	$[CON, \zeta] \Rightarrow [DEC, \eta](C1.1)$
$CASE2$	$g(y, CON) = \{\zeta\}$	$\eta \in g(y, DEC)$	$[CON, \zeta] \Rightarrow [DEC, \eta](C2.1)$ $[CON, \zeta] \Rightarrow [DEC, \eta'](C2.2)$
$CASE3$	$g(y, CON) = \{\zeta\}$	$\eta \notin g(y, DEC)$	$[CON, \zeta] \Rightarrow [DEC, \eta'](C3.1)$
$CASE4$	$\zeta \in g(y, CON)$	$g(y, DEC) = \{\eta\}$	$[CON, \zeta] \Rightarrow [DEC, \eta](C4.1)$ $[CON, \zeta'] \Rightarrow [DEC, \eta](C4.2)$
$CASE5$	$\zeta \in g(y, CON)$	$\eta \in g(y, DEC)$	$[CON, \zeta] \Rightarrow [DEC, \eta](C5.1)$ $[CON, \zeta] \Rightarrow [DEC, \eta'](C5.2)$ $[CON, \zeta'] \Rightarrow [DEC, \eta](C5.3)$ $[CON, \zeta'] \Rightarrow [DEC, \eta'](C5.4)$
$CASE6$	$\zeta \in g(y, CON)$	$\eta \notin g(y, DEC)$	$[CON, \zeta] \Rightarrow [DEC, \eta'](C6.1)$ $[CON, \zeta'] \Rightarrow [DEC, \eta'](C6.2)$
$CASE7$	$\zeta \notin g(y, CON)$	Any	$[CON, \zeta'] \Rightarrow Decision(C7.1)$

Now, we revise the definition of inf and sup information in the previous section. We handled both inf and sup information for every object x. However, in the subsequent sections it is enough to handle minimum and maximum sets of an equivalence class defined by a descriptor $[ATR, val]$. This revision is very simple, and this revision reduces the manipulation of each calculation.

Definition 1. For each descriptor $[ATR, val](=[\{A_1, \cdots, A_k\}, (\zeta_1, \cdots, \zeta_k)], (k \geq 1)$) in a NIS, $Descinf$ and $Descsup$ are defined as follows:

(1) $Descinf([A_i,\zeta_i])=\{x \in OB|PT(x,\{A_i\})=\{\zeta_i\}\}=\{x{\in}OB|g(x,\{A_i\})=\{\zeta_i\}\}$.
(2) $Descinf([ATR,val])=Descinf(\wedge_i[A_i,\zeta_i])=\cap_i Descinf([A_i,\zeta_i])$.
(3) $Descsup([A_i,\zeta_i])=\{x \in OB|\zeta_i \in PT(x,\{A_i\})\}=\{x \in OB|\zeta_i \in g(x,\{A_i\})\}$.
(4) $Descsup([ATR,val])=Descsup(\wedge_i[A_i,\zeta_i])=\cap_i Descsup([A_i,\zeta_i])$.

The definition of *Descinf* requires that every element in this set is definite. Even though the definition of *Descsup* is the same as *sup*, we employ the $Descsup([ATR,\zeta])$ notation due to the $Descinf([ATR,\zeta])$ notation. Clearly, $Descinf([CON,\zeta])$ is a set of objects belonging to either $CASE$ 1, 2 or 3 in Table 5, and $Descsup([CON,\zeta])$ is a set of objects belonging to either $CASE$ 1 to $CASE$ 6. $Descsup([CON,\zeta]) - Descinf([CON,\zeta])$ is a set of objects belonging to either $CASE$ 4, 5 or 6.

Proposition 3. Let $|X|$ denote the cardinality of a set X. In Table 6, the *support* value of $\tau^x : [CON,\zeta] \Rightarrow [DEC,\eta]$ from x is minimum.

If τ^x is definite, namely τ^x belongs to $CASE$ 1,

$$minsupp(\tau^x)=|Descinf([CON,\zeta]) \cap Descinf([DEC,\eta])|/|OB|.$$

If τ^x is indefinite, namely τ^x does not belong to $CASE$ 1,

$$minsupp(\tau^x)=(|Descinf([CON,\zeta]) \cap Descinf([DEC,\eta])| + 1)/|OB|.$$

Proof. This selection of attribute values in a NIS excludes every $[CON,\zeta] \Rightarrow [DEC,\eta]$ from object $y \neq x$. In reality, we remove (C2.1), (C4.1) and (C5.1) from Table 5. Therefore, the *support* value of τ^x is minimum in a derived DIS with such selections of attribute values. If τ^x is definite, object x is in a set $Descinf([CON,\zeta]){\cap}Descinf([DEC,\eta])$. Otherwise, τ^x belongs to either (C2.1), (C4.1) or (C5.1). Thus, it is necessary to add 1 to the numerator.

Proposition 4. Table 7 is a part of Table 5. In Table 7, the *accuracy* value of $\tau^x : [CON,\zeta] \Rightarrow [DEC,\eta]$ from x is minimum. Let $OUTACC$ denote

$$[Descsup([CON,\zeta]) - Descinf([CON,\zeta])] - Descinf([DEC,\eta]).$$

Table 6. Selections from Table 5. These selections make the *support* value of $[CON,\zeta] \Rightarrow [DEC,\eta]$ minimum.

	$Condition : CON$	$Decision : DEC$	$Selection$
$CASE1$	$g(y,CON)=\{\zeta\}$	$g(y,DEC)=\{\eta\}$	$[CON,\zeta] \Rightarrow [DEC,\eta](C1.1)$
$CASE2$	$g(y,CON)=\{\zeta\}$	$\eta \in g(y,DEC)$	$[CON,\zeta] \Rightarrow [DEC,\eta'](C2.2)$
$CASE3$	$g(y,CON)=\{\zeta\}$	$\eta \notin g(y,DEC)$	$[CON,\zeta] \Rightarrow [DEC,\eta'](C3.1)$
$CASE4$	$\zeta \in g(y,CON)$	$g(y,DEC)=\{\eta\}$	$[CON,\zeta'] \Rightarrow [DEC,\eta](C4.2)$
$CASE5$	$\zeta \in g(y,CON)$	$\eta \in g(y,DEC)$	$[CON,\zeta] \Rightarrow [DEC,\eta'](C5.2)$ $[CON,\zeta'] \Rightarrow [DEC,\eta](C5.3)$ $[CON,\zeta'] \Rightarrow [DEC,\eta'](C5.4)$
$CASE6$	$\zeta \in g(y,CON)$	$\eta \notin g(y,DEC)$	$[CON,\zeta] \Rightarrow [DEC,\eta'](C6.1)$ $[CON,\zeta'] \Rightarrow [DEC,\eta'](C6.2)$
$CASE7$	$\zeta \notin g(y,CON)$	Any	$[CON,\zeta'] \Rightarrow Decision(C7.1)$

Table 7. Selections from Table 5. These selections make the *accuracy* value of $[CON,\zeta] \Rightarrow [DEC,\eta]$ minimum.

	$Condition : CON$	$Decision : DEC$	$Selection$
$CASE1$	$g(y,CON) = \{\zeta\}$	$g(y,DEC) = \{\eta\}$	$[CON,\zeta] \Rightarrow [DEC,\eta](C1.1)$
$CASE2$	$g(y,CON) = \{\zeta\}$	$\eta \in g(y,DEC)$	$[CON,\zeta] \Rightarrow [DEC,\eta'](C2.2)$
$CASE3$	$g(y,CON) = \{\zeta\}$	$\eta \notin g(y,DEC)$	$[CON,\zeta] \Rightarrow [DEC,\eta'](C3.1)$
$CASE4$	$\zeta \in g(y,CON)$	$g(y,DEC) = \{\eta\}$	$[CON,\zeta'] \Rightarrow [DEC,\eta](C4.2)$
$CASE5$	$\zeta \in g(y,CON)$	$\eta \in g(y,DEC)$	$[CON,\zeta] \Rightarrow [DEC,\eta'](C5.2)$
$CASE6$	$\zeta \in g(y,CON)$	$\eta \notin g(y,DEC)$	$[CON,\zeta] \Rightarrow [DEC,\eta'](C6.1)$
$CASE7$	$\zeta \notin g(y,CON)$	Any	$[CON,\zeta'] \Rightarrow Decision(C7.1)$

If τ^x is definite,

$$minacc(\tau^x)=\frac{|Descinf([CON,\zeta]) \cap Descinf([DEC,\eta])|}{|Descinf([CON,\zeta])|+|OUTACC|}.$$

If τ^x is indefinite,

$$minacc(\tau^x)=\frac{|Descinf([CON,\zeta]) \cap Descinf([DEC,\eta])|+1}{|Descinf([CON,\zeta]) \cup \{x\}|+|OUTACC-\{x\}|}.$$

Proof. Since $m/n \leq (m+k)/(n+k)$ $(0 \leq m \leq n,\ n \neq 0,\ k > 0)$ holds, we excludes every $[CON,\zeta] \Rightarrow [DEC,\eta]$ from object $y \neq x$. We select possible implications $[CON,\zeta] \Rightarrow [DEC,\eta']$, which increase the denominator. The *accuracy* value of τ^x is minimum in a derived DIS with such selection of attribute values. The set $OUTACC$ defines objects in either $CASE$ 5 or $CASE$ 6. As for $CASE$ 4 and $CASE$ 7, the condition part is not $[CON,\zeta]$. Therefore, we can omit such implications for calculating $minacc(\tau^x)$. If τ^x is definite, the numerator is $|Descinf([CON,\zeta]) \cap Descinf([DEC,\eta])|$ and the denominator is $|Descinf([CON,\zeta])|+|OUTACC|$. If τ^x is indefinite, τ^x belongs to either (C2.1), (C4.1) or (C5.1). The denominator is $|Descinf([CON,\zeta]) \cup \{x\}| + |OUTACC - \{x\}|$ in every case, and the numerator is $|Descinf([CON,\zeta]) \cap Descinf\ ([DEC,\eta])|+1$.

Theorem 5. For a NIS, let us consider a possible implication τ^x:$[CON,\zeta] \Rightarrow [DEC,\eta] \in PI(x,CON,DEC)$. Let $SUPP_{min}$=$\{\varphi|\varphi$ is a derived DIS from NIS, and $support(\tau^x)$ is minimum in $\varphi\}$. Then, $accuracy(\tau^x)$ is minimum in some $\varphi \in SUPP_{min}$.

Proof. Table 7 is a special case of Table 6. Namely, in $CASE$ 5 of Table 6, either (C5.2), (C5.3) or (C5.4) may hold. In $CASE$ 6 of Table 6, either (C6.1) or (C6.2) may hold. In every selection, the minimum *support* value is the same. In Table 7, (C5.2) in $CASE$ 5 and (C6.1) in $CASE$ 6 are selected.

Theorem 5 assures that there exists a derived DIS, where both $support(\tau^x)$ and $accuracy(\tau^x)$ are minimum. DIS_{worst} denotes such a derived DIS, and we name

Table 8. Selections from Table 5. These selections make the *support* and *accuracy* values of $[CON, \zeta] \Rightarrow [DEC, \eta]$ maximum.

	$Condition(CON)$	$Decision(DEC)$	$Selection$
$CASE1$	$g(y, CON) = \{\zeta\}$	$g(y, DEC) = \{\eta\}$	$[CON, \zeta] \Rightarrow [DEC, \eta](C1.1)$
$CASE2$	$g(y, CON) = \{\zeta\}$	$\eta \in g(y, DEC)$	$[CON, \zeta] \Rightarrow [DEC, \eta](C2.1)$
$CASE3$	$g(y, CON) = \{\zeta\}$	$\eta \notin g(y, DEC)$	$[CON, \zeta] \Rightarrow [DEC, \eta'](C3.1)$
$CASE4$	$\zeta \in g(y, CON)$	$g(y, DEC) = \{\eta\}$	$[CON, \zeta] \Rightarrow [DEC, \eta](C4.1)$
$CASE5$	$\zeta \in g(y, CON)$	$\eta \in g(y, DEC)$	$[CON, \zeta] \Rightarrow [DEC, \eta](C5.1)$
$CASE6$	$\zeta \in g(y, CON)$	$\eta \notin g(y, DEC)$	$[CON, \zeta'] \Rightarrow [DEC, \eta'](C6.2)$
$CASE7$	$\zeta \notin g(y, CON)$	Any	$[CON, \zeta'] \Rightarrow Decision(C7.1)$

DIS_{worst} a *derived DIS* with the *worst condition* for τ^x. This is an important property for Problem 3 in the subsequent section.

3.4 Effective Calculation of Maximum Support and Maximum Accuracy

In this subsection, we show an effective method to calculate $maxsupp(\tau^x)$ and $maxacc(\tau^x)$ based on $Descinf$ and $Descsup$. The following can be proved according the same manner as Proposition 3, 4 and Theorem 5. A derived DIS defined in Table 8 makes both *support* and *accuracy* maximum.

Proposition 6. For $\tau^x : [CON, \zeta] \Rightarrow [DEC, \eta]$ from x, the following holds.

$$maxsupp(\tau^x) = |Descsup([CON, \zeta]) \cap Descsup([DEC, \eta])|/|OB|.$$

Proposition 7. For $\tau^x : [CON, \zeta] \Rightarrow [DEC, \eta]$ from x, let $INACC$ denote

$[Descsup([CON, \zeta]) - Descinf([CON, \zeta])] \cap Descsup\ ([DEC, \eta])$.

If τ^x is definite,

$$maxacc(\tau^x) = \frac{|Descinf([CON,\zeta]) \cap Descsup([DEC,\eta])| + |INACC|}{|Descinf([CON,\zeta])| + |INACC|}.$$

If τ^x is indefinite,

$$maxacc(\tau^x) = \frac{|Descinf([CON,\zeta]) \cap Descsup([DEC,\eta]) - \{x\}| + |INACC - \{x\}| + 1}{|Descinf([CON,\zeta]) \cup \{x\}| + |INACC - \{x\}|}.$$

Theorem 8. For a NIS, let us consider a possible implication $\tau^x:[CON, \zeta] \Rightarrow [DEC, \eta] \in PI(x, CON, DEC)$. Let $SUPP_{max} = \{\varphi | \varphi$ is a derived DIS from NIS, and $support(\tau^x)$ is maximum in $\varphi\}$. Then, $accuracy(\tau^x)$ is maximum in some $\varphi \in SUPP_{max}$.

Theorem 8 assures that there exists a derived DIS, where both $support(\tau^x)$ and $accuracy(\tau^x)$ are maximum. DIS_{best} denotes such a derived DIS, and we name DIS_{best} a *derived DIS* with the *best condition* for τ^x. This is also an important property for Problem 4 in the subsequent section.

4 Rule Generation by New Criteria in Non-deterministic Information Systems

This section applies Proposition 3, 4, 6, 7 and Theorem 5, 8 to rule generation in $NISs$.

4.1 Rules by the Criteria in Deterministic Information Systems

In $DISs$, rule generation by the criteria is often defined as the following.

Problem 2. In a table or a DIS, find every implication τ that $support(\tau) \geq \alpha$ and $accuracy(\tau) \geq \beta$ for given α and β $(0 < \alpha, \beta \leq 1)$.

For solving this problem, *Apriori* algorithm was proposed by Agrawal [1, 2]. In this framework, *association rules* in *transaction data* are obtained. The application of the *large item set* is the key point in *Apriori* algorithm. This Problem 2 has also been considered in [22, 36, 38].

4.2 Rules by New Criteria and Two Strategies in Non-deterministic Information Systems

Now, we extend Problem 2 to Problem 3 and Problem 4 in the following.

Problem 3 (Rule Generation by Lower Approximation Strategy). For a NIS, let $CON \subseteq AT$ and $DEC \subseteq AT$ be condition attributes and the decision attribute, respectively. Find every possible implication $\tau^x : [CON, \zeta] \Rightarrow [DEC, \eta]$ satisfying $minsupp(\tau^x) \geq \alpha$ and $minacc(\tau^x) \geq \beta$ for given α and β $(0 < \alpha, \beta \leq 1)$.

Problem 4 (Rule Generation by Upper Approximation Strategy). For a NIS, let $CON \subseteq AT$ and $DEC \subseteq AT$ be condition attributes and the decision attribute, respectively. Find every possible implication $\tau^x : [CON, \zeta] \Rightarrow [DEC, \eta]$ satisfying $maxsupp(\tau^x) \geq \alpha$ and $maxacc(\tau^x) \geq \beta$ for given α and β $(0 < \alpha, \beta \leq 1)$.

It is necessary to remark that both $minsupp(\tau^x)$ and $minacc(\tau^x)$ are defined over $DD(\tau^x, x, CON, DEC)$. For definite τ^x, $DD(\tau^x, x, CON, DEC)$ is equal to all derived $DISs$. However for indefinite τ^x, $DD(\tau^x, x, CON, DEC)$ is not equal to all derived $DISs$, and $minsupp(\tau^x)$=0 and $minacc(\tau^x)$=0 may hold. This may be an important issue in *lower approximation strategy*. However in this paper, we employ a set $DD(\tau^x, x, CON, DEC)$ instead of all derived $DISs$. As for *upper approximation strategy*, $maxsupp(\tau^x)$ and $maxacc(\tau^x)$ over $DD(\tau^x, x, CON,\ DEC)$ are the same as $maxsupp(\tau^x)$ and $maxacc(\tau^x)$ over all derived $DISs$. We employed terms Min-Max and Max-Max strategies in [31, 32]. According to rough sets based concept, we rename these terms *lower approximation strategy* and *upper approximation strategy*, respectively.

Next Proposition 9 clarifies the relation between certain rules, possible rules and rules by new criteria.

Proposition 9. For a possible implication τ^x, the following holds.

(1) τ^x is a certain rule in Section 2.1, if and only if τ^x is definite and $minacc(\tau^x)$=1.
(2) τ^x is a possible rule in Section 2.1, if and only if $maxacc(\tau^x)$=1.

The concept of consistency defines certain and possible rules, therefore there is no definition about *support*. In certain rule generation, we often have a possible implication whose $minacc(\tau^x)$=1 and $minsupp(\tau^x)$ is quite small. Proposition 10, 11 and 12 clarify the properties of rule generation.

Proposition 10. For a given α and β $(0 < \alpha, \beta \leq 1)$, let $Rule(\alpha, \beta, LA)$ denote a set of rules defined by *lower approximation strategy* with α and β, and let $Rule(\alpha, \beta, UA)$ denote a set of rules defined by *upper approximation strategy* with α and β. Then, $Rule(\alpha, \beta, LA) \subseteq Rule(\alpha, \beta, UA)$ holds.

Proposition 11. The following, which are related to a possible implication $\tau^x : [CON, \zeta] \Rightarrow [DEC, \eta]$, are equivalent.

(1) τ^x is obtained according to *lower approximation strategy*, namely $minsupp(\tau^x) \geq \alpha$ and $minacc(\tau^x) \geq \beta$.
(2) $support(\tau^x) \geq \alpha$ and $accuracy(\tau^x) \geq \beta$ in each $\varphi \in DD(\tau^x, x, CON, DEC)$.
(3) In a derived DIS_{worst} defined in Table 7, $support(\tau^x) \geq \alpha$ and $accuracy(\tau^x) \geq \beta$ hold.

Proof. For each $\varphi \in DD(\tau^x, x, CON, DEC)$, $support(\tau^x) \geq minsupp(\tau^x)$ and $accuracy(\tau^x) \geq minacc(\tau^x)$ hold, therefore (1) and (2) are equivalent. According to Theorem 5, a derived DIS_{worst} (depending upon τ^x) defined in Table 7 assigns minimum values to both $support(\tau^x)$ and $accuracy(\tau^x)$. Thus, (1) and (3) are equivalent.

Proposition 12. The following, which are related to a possible implication $\tau^x : [CON, \zeta] \Rightarrow [DEC, \eta]$, are equivalent.

(1) τ^x is obtained according to *upper approximation strategy*, namely $maxsupp(\tau^x) \geq \alpha$ and $maxacc(\tau^x) \geq \beta$.
(2) $support(\tau^x) \geq \alpha$ and $accuracy(\tau^x) \geq \beta$ in a $\varphi \in DD(\tau^x, x, CON, DEC)$.
(3) In a derived DIS_{best} defined in Table 8, $support(\tau^x) \geq \alpha$ and $accuracy(\tau^x) \geq \beta$ hold.

Proof: For each $\varphi \in DD(\tau^x, x, CON, DEC)$, $support(\tau^x) \leq maxsupp(\tau^x)$ and $accuracy(\tau^x) \leq maxacc(\tau^x)$ hold. According to Theorem 8, a derived DIS_{best} (depending upon τ^x) defined in Table 8 assigns maximum values to both $support(\tau^x)$ and $accuracy(\tau^x)$. In this DIS_{best}, $maxsupp(\tau^x)$=$support(\tau^x)$ and $maxacc(\tau^x)$=$accuracy(\tau^x)$ hold. Thus, (1), (2) and (3) are equivalent.

Due to Proposition 10, 11 and 12, $Rule(\alpha, \beta, LA)$ defines a set of possible implications in a DIS_{worst}, and $Rule(\alpha, \beta, UA)$ defines a set of possible implications in a DIS_{best}. This implies that we do not have to examine each derived DIS in

$DD(\tau^x, x, CON, DEC)$, but we have only to examine a DIS_{worst} for the *lower approximation strategy* and a DIS_{best} for the *upper approximation strategy.*

4.3 Extended Apriori Algorithms for Two Strategies and A Simulation

This subsection proposes two extended *Apriori* algorithms in Algorithm 1 and 2. In $DISs$, $Descinf([A,\zeta])$=$Descsup([A,\zeta])$ holds, however $Descinf([A,\zeta]) \subseteq Descsup([A,\zeta])$ holds in $NISs$. *Apriori* algorithm handles transaction data, and employs the sequential search for obtaining large item sets [1, 2]. In $DISs$, we employ the manipulation of $Descinf$ and $Descsup$ instead of the sequential search. According to this manipulation, we obtain the minimum set and maximum set of an equivalence class. Then, we calculate $minsupp(\tau^x)$ and $minacc(\tau^x)$ by using $Descinf$ and $Descsup$. The rest is almost the same as *Apriori* algorithm.

Now, we show an example, which simulates Algorithm 1.

Algorithm 1. Extended Apriori Algorithm for Lower Approximation Strategy

```
Input  : A $NIS$, a decision attribute $DEC$, threshold value $\alpha$ and $\beta$.
Output: Every rule defined by lower approximation strategy.
for ($every\ A \in AT$) do
  | Generate $Descinf([A,\zeta])$ and $Descsup([A,\zeta])$;
end
For the condition $minsupp(\tau^x)=|SET|/|OB| \geq \alpha$, obtain the number $NUM$ of
    elements in $SET$;
Generate a set $CANDIDATE(1)$, which consists of descriptors $[A,\zeta_A]$
    satisfying either ($CASE$ A) or ($CASE$ B) in the following;
    ($CASE$ A) $|Descinf([A,\zeta_A])| \geq NUM$,
    ($CASE$ B) $|Descinf([A,\zeta_A])|=(NUM-1)$ and
                  $(Descsup([A,\zeta_A]) - Descinf([A,\zeta_A])) \neq \{\}$.
Generate a set $CANDIDATE(2)$ according to the following procedures;
    (Proc 2-1) For every $[A,\zeta_A]$ and $[DEC,\zeta_{DEC}]$ $(A \neq DEC)$ in
        $CANDIDATE(1)$, generate a new descriptor $[\{A,DEC\},(\zeta_A,\zeta_{DEC})]$;
    (Proc 2-2) Examine condition ($CASE$ A) and ($CASE$ B) for each
        $[\{A,DEC\},(\zeta_A,\zeta_{DEC})]$;
        If either ($CASE$ A) or ($CASE$ B) holds and $minacc(\tau) \geq \beta$
          display $\tau : [A,\zeta_A] \Rightarrow [DEC,\zeta_{DEC}]$ as a rule;
        If either ($CASE$ A) or ($CASE$ B) holds and $minacc(\tau) < \beta$,
          add this descriptor to $CANDIDATE(2)$;
Assign 2 to $n$;
while $CANDIDATE(n) \neq \{\}$ do
  | Generate $CANDIDATE(n+1)$ according to the following procedures;
  |   (Proc 3-1) For $DESC_1$ and $DESC_2$ ($[DEC,\zeta_{DEC}] \in DESC_1 \cap DESC_2$ )
  |       in $CANDIDATE(n)$, generate a new descriptor by using a
  |       conjunction of $DESC_1 \wedge DESC_2$;
  |   (Proc 3-2) Examine the same procedure as (Proc 2-2).
  | Assign $n+1$ to $n$;
end
```

```
Algorithm 2. Extended Apriori Algorithm for Upper Approximation Strategy
Input  : A $NIS$, a decision attribute $DEC$, threshold value $\alpha$ and $\beta$.
Output: Every rule defined by upper approximation strategy.
Algorithm 2 is proposed as Algorithm 1 with the following two revisions :
  1. ($CASE$ A) and ($CASE$ B) in Algorithm 1 are replaced with ($CASE$ C).
     ($CASE$ C) $|Descsup([A, \zeta_A])| \geq NUM$.
  2. $minacc(\tau)$ in Algorithm 1 is replaced with $maxacc(\tau)$.
```

Example 3. Let us consider $Descinf$ and $Descsup$, which are obtained from NIS_2 in Table 9, and let us consider Problem 3. We set α=0.3, β=0.8, condition attribute $CON \subseteq \{P, Q, R, S\}$ and decision attribute $DEC=\{T\}$. Since $|OB|$=5 and $minsupp(\tau)=|SET|/5 \geq 0.3$, $|SET| \geq 2$ must hold. According to Table 10, we generate Table 11 satisfying either ($CASE$ A) or ($CASE$ B) in the following:

($CASE$ A) $|Descinf([A, \zeta_A] \wedge [T, \eta])| \geq 2$ $(A \in \{P, Q, R, S\})$.
($CASE$ B) $|Descinf([A, \zeta_A] \wedge [T, \eta])|=1$ and $Descsup([A, \zeta_A] \wedge [T, \eta]) - Descinf([A, \zeta_A] \wedge [T, \eta]) \neq \{\}$ $(A \in \{P, Q, R, S\})$.

Table 9. A Table of NIS_2

OB	P	Q	R	S	T
1	$\{3\}$	$\{1,3\}$	$\{3\}$	$\{2\}$	$\{3\}$
2	$\{2\}$	$\{2,3\}$	$\{1,3\}$	$\{1,3\}$	$\{2\}$
3	$\{1,2\}$	$\{2\}$	$\{1,2\}$	$\{3\}$	$\{1\}$
4	$\{1\}$	$\{3\}$	$\{3\}$	$\{2,3\}$	$\{1,2,3\}$
5	$\{3\}$	$\{1\}$	$\{1,2\}$	$\{3\}$	$\{3\}$

Table 10. $Descinf$ and $Descsup$ information in Table 9

	$[P,1]$	$[P,2]$	$[P,3]$	$[Q,1]$	$[Q,2]$	$[Q,3]$	$[R,1]$	$[R,2]$	$[R,3]$
$Descinf$	$\{4\}$	$\{2\}$	$\{1,5\}$	$\{5\}$	$\{3\}$	$\{4\}$	$\{\}$	$\{\}$	$\{1,4\}$
$Descsup$	$\{3,4\}$	$\{2,3\}$	$\{1,5\}$	$\{1,5\}$	$\{2,3\}$	$\{1,2,4\}$	$\{2,3,5\}$	$\{3,5\}$	$\{1,2,4\}$

	$[S,1]$	$[S,2]$	$[S,3]$	$[T,1]$	$[T,2]$	$[T,3]$
$Descinf$	$\{\}$	$\{1\}$	$\{3,5\}$	$\{3\}$	$\{2\}$	$\{1,5\}$
$Descsup$	$\{2\}$	$\{1,4\}$	$\{2,3,4,5\}$	$\{3,4\}$	$\{2,4\}$	$\{1,4,5\}$

Table 11. Conjunctions of descriptors satisfying either ($CASE$ A) or ($CASE$ B) in Table 10

	$[P,3] \wedge [T,3]$	$[Q,1] \wedge [T,3]$	$[R,3] \wedge [T,3]$	$[S,2] \wedge [T,3]$	$[S,3] \wedge [T,1]$	$[S,3] \wedge [T,3]$
$Descinf$	$\{1,5\}$	$\{5\}$	$\{1\}$	$\{1\}$	$\{3\}$	$\{5\}$
$Descsup$	$\{1,5\}$	$\{1,5\}$	$\{1,4\}$	$\{1,4\}$	$\{3,4\}$	$\{4,5\}$

The conjunction $[P,3] \wedge [T,3]$ in Table 11 means an implication $\tau_3^1, \tau_3^5 : [P,3] \Rightarrow [T,3]$. Because $Descsup([P,3] \wedge [T,3])$=$\{1,5\}$ holds, τ_3^1 and τ_3^5 come from object 1 and 5, respectively. Since $1,5 \in Descinf([P,3] \wedge [T,3])$ holds, $minsupp(\tau_3^1)$= $minsupp(\tau_3^5)$=$|\{1,5\}|/5$=0.4 holds. Then, the conjunction $[Q,1] \wedge [T,3]$ in Table 11 means an implication $\tau_4^1, \tau_4^5 : [Q,1] \Rightarrow [T,3]$. Since $5 \in Descinf([Q,1] \wedge [T,3])$ holds, $minsupp(\tau_4^5)$=$|\{5\}|/5$=0.2 holds. On the other hand, $1 \in Descsup([Q,1] \wedge [T,3]) - Descinf([Q,1] \wedge [T,3])$ holds, so $minsupp(\tau_4^1)$=$(|\{5\}|+1)/5$=0.4 holds in object 1. According to this consideration, we obtain the candidates of rules, which satisfy $minsupp(\tau^x) \geq 0.3$, as follows:

$\tau_3^1, \tau_3^5 : [P,3] \Rightarrow [T,3]$, $\tau_4^1 : [Q,1] \Rightarrow [T,3]$, $\tau_5^4 : [R,3] \Rightarrow [T,3]$,
$\tau_6^4 : [S,2] \Rightarrow [T,3]$, $\tau_7^4 : [S,3] \Rightarrow [T,1]$, $\tau_8^4 : [S,3] \Rightarrow [T,3]$.

For these candidates, we examine each $minacc(\tau^x)$ according to Proposition 4. For τ_3^1 and τ_3^5, $Descsup([P,3])$=$\{1,5\}$, $Descinf([P,3])$=$\{1,5\}$, $Descinf([P,3] \wedge [T,3])$=$\{1,5\}$ and $OUTACC$=$[\{1,5\}-\{1,5\}]-\{1,5\}$=$\{\}$. Since $1,5 \in Descinf([P,3] \wedge [T,3])$ holds, $minacc(\tau_3^1)$= $minacc(\tau_3^5)$=$|\{1,5\}|/(|\{1,5\}|+|\{\}|)$=1 is derived. For $\tau_7^4 : [S,3] \Rightarrow [T,1]$, $Descsup([S,3])$=$\{2,3,4,5\}$, $Descinf([S,3])$=$\{3,5\}$, $Descinf([S,3] \wedge [T,1])$=$\{3\}$, $Descsup([S,3] \wedge [T,1])$= $\{3,4\}$ and $OUTACC$=$[\{2,3,4,5\}-\{3,5\}]-\{3\}$=$\{2,4\}$ holds, so $minacc(\tau_7^4)$=$(|\{3\}|+1)/(|\{3,5\} \cup \{4\}|+|\{2,4\}-\{4\}|)$=0.5 is derived. In this way, we obtain three rules satisfying $minsupp(\tau^x) \geq 0.3$ and $minacc(\tau^x) \geq 0.8$ in the following:

$\tau_3^1, \tau_3^5 : [P,3] \Rightarrow [T,3]$ ($minsupp$=0.4, $minacc$=1),
$\tau_4^1 : [Q,1] \Rightarrow [T,3]$ ($minsupp$=0.4, $minacc$=1),
$\tau_6^4 : [S,2] \Rightarrow [T,3]$ ($minsupp$=0.4, $minacc$=1).

Any possible implication including $[R,3] \wedge [T,3]$ does not satisfy $minsupp(\tau^x) \geq 0.3$. As for $[S,3] \wedge [T,1]$ and $[S,3] \wedge [T,3]$, the same results hold.

The following shows a real execution on Example 3.

```
% ./nis_apriori
version 1.2.8
File Name:'nis2.dat'
========================================
Lower Approximation Strategy
========================================
CAN(1)=[P,1],[P,2],[P,3],[Q,1],[Q,2],[Q,3],[R,3],[S,2],[S,3],[T,1],
[T,2],[T,3](12)
CAN(2)=[S,3][T,1](<DEF>0.250,<INDEF>0.500),[P,3][T,3](<DEF>1.000,
<INDEF>1.000),[Q,1][T,3](<DEF>1.000,<INDEF>1.000),[R,3][T,3](<DEF>0.333,
<INDEF>0.667),[S,2][T,3](<DEF>0.500,<INDEF>1.000),[S,3][T,3](<DEF>0.250,
<INDEF>0.500)(6)
========== OBTAINED RULE ==========
[P,3]=>[T,3](minsupp<DEF>=0.400,minsupp<INDEF>=0.400,minacc<DEF>=1.000,
minacc<INDEF>=1.000) (<DEF>from 1,5) (<INDEF>from )
[Q,1]=>[T,3](minsupp<DEF>=0.200,minsupp<INDEF>=0.400,minacc<DEF>=1.000,
minacc<INDEF>=1.000) (<DEF>from ) (<INDEF>from 1)
```

```
[S,2]=>[T,3](minsupp<DEF>=0.200,minsupp<INDEF>=0.400,minacc<DEF>=0.500,
minacc<INDEF>=1.000) (<DEF>from ) (<INDEF>from 4)
EXEC_TIME=0.0000000000(sec)

========================================
Upper Approximation Strategy
========================================
CAN(1)=[P,1],[P,2],[P,3],[Q,1],[Q,2],[Q,3],[R,3],[S,2],[S,3],[T,1],
[T,2],[T,3](12)
CAN(2)=[S,3][T,1](<DEF>0.667,<INDEF>0.667),[P,3][T,3](<DEF>1.000,
<INDEF>1.000),[Q,1][T,3](<DEF>1.000,<INDEF>1.000),[R,3][T,3](<DEF>1.000,
<INDEF>1.000),[S,2][T,3](<DEF>1.000,<INDEF>1.000),[S,3][T,3](<DEF>0.667,
<INDEF>0.667)(6)
========== OBTAINED RULE ==========
[P,3]=>[T,3](maxsupp<DEF>=0.400,maxsupp<INDEF>=0.400,maxacc<DEF>=1.000,
maxacc<INDEF>=1.000) (<DEF>from 1,5) (<INDEF>from )
[Q,1]=>[T,3](maxsupp<DEF>=0.400,maxsupp<INDEF>=0.400,maxacc<DEF>=1.000,
maxacc<INDEF>=1.000) (<DEF>from 5) (<INDEF>from 1)
[R,3]=>[T,3](maxsupp<DEF>=0.400,maxsupp<INDEF>=0.400,maxacc<DEF>=1.000,
maxacc<INDEF>=1.000) (<DEF>from 1) (<INDEF>from 4)
[S,2]=>[T,3](maxsupp<DEF>=0.400,maxsupp<INDEF>=0.400,maxacc<DEF>=1.000,
maxacc<INDEF>=1.000) (<DEF>from 1) (<INDEF>from 4)
EXEC_TIME=0.0000000000(sec)
```

According to this execution, we know

$Rule(0.3, 0.8, LA)=\{[P, 3] \Rightarrow [T, 3], [Q, 1] \Rightarrow [T, 3], [S, 2] \Rightarrow [T, 3]\}$,
$Rule(0.3, 0.8, UA)=\{[P, 3] \Rightarrow [T, 3], [Q, 1] \Rightarrow [T, 3], [S, 2] \Rightarrow [T, 3], [R, 3] \Rightarrow [T, 3]\}$.

The possible implication $[R, 3] \Rightarrow [T, 3] \in Rule(0.3, 0.8, UA) - Rule(0.3, 0.8, LA)$ depends upon the information incompleteness. This can not be obtained by the *lower approximation strategy*, but this can be obtained by the *upper approximation strategy*.

4.4 Main Program for Lower Approximation Strategy

A program *nis_apriori* is implemented on a Windows PC with Pentium 4 (3.40 GHz), and it consists of about 1700 lines in C. This *nis_apriori* mainly consists of two parts, i.e., a part for *lower approximation strategy* and a part for *upper approximation strategy*.

As for *lower approximation strategy*, a function *GenRuleByLA*() (Generate Rules By LA strategy) is coded.

```
GenRuleByLA(table.obj,table.att,table.kosuval,table.con_num,
  table.dec_num,table.con,table.dec,thresh,minacc_thresh);
```

In *GenRuleByLA*(), a function *GenCandByLA*() is called, and generates a candidate $CANDIDATE(n)$.

```
GenCandByLA(desc,cand,conj_num_max,ob,at,desc_num,c_num,
  d_num,co,de,thr,minacc_thr);
```

At the same time, $minsupp(\tau)$ and $minacc(\tau)$ are calculated according to Proposition 3 and 4. As for *upper approximation strategy*, the similar functions are implemented.

5 Computational Issues in Algorithm 1

This section focuses on the computational complexity of Algorithm 1. As for Algorithm 2, the result is almost the same as Algorithm 1.

5.1 A Simple Method for Lower Approximation Strategy

Generally, a possible implication τ^x depends upon the number of derived $DISs$, i.e., $\prod_{x \in OB, A \in AT} |g(x,A)|$, and condition attributes CON ($CON \subseteq 2^{AT-DEC}$). Furthermore, $minsupp(\tau^x)$ and $minacc(\tau^x)$ depend on $DD(\tau^x, x, CON, DEC)$, whose number of elements is $\prod_{A \in CON, B \in DEC, x \neq y} |g(y,A)||g(y,B)|$. Therefore, it will be impossible to employ a simple method that we sequentially pick up every possible implication τ^x and sequentially examine $minsupp(\tau^x)$ and $minacc(\tau^x)$.

5.2 Complexity on Extended Apriori Algorithm for Lower Approximation Strategy

In order to solve this computational issue, we focus on descriptors $[A, \zeta]$ ($A \in AT$, $\zeta \in VAL_A$). The number of all descriptors is usually very small. Furthermore, Proposition 3 and 4 show us the methods to calculate $minsupp(\tau^x)$ and $minacc(\tau^x)$. These methods do not depend upon the number of element in $DD(\tau^x, x, CON, DEC)$.

Now, we analyze each step in Algorithm 1.

(STEP 1) (Generation of $Descinf$, $Descsup$ and $CANDIDATE(1)$)
We first prepare two arrays $Descinf_{A,val}[]$ and $Descsup_{A,val}[]$ for each $val \in VAL_A$ ($A \in AT$). For each object $x \in OB$, we apply (1) and (2) in the following:

(1) If $g(x,A)=\{val\}$, add x to $Descinf_{A,val}[]$ and $Descsup_{A,val}[]$.
(2) If $g(x,A) \neq \{val\}$ and $val \in g(x,A)$, add x to $Descsup_{A,val}[]$.

Then, all descriptors satisfying either ($CASE$ A) or ($CASE$ B) in Algorithm 1 are added to $CANDIDATE(1)$. For each $A \in AT$, this procedure is applied, and the complexity depends upon $|OB| \times |AT|$.

(STEP 2) (Generation of $CANDIDATE(2)$)
For each $[A, val_A], [DEC, val_{DEC}] \in CANDIDATE(1)$, we produce $[A, val_A] \wedge [DEC, val_{DEC}]$, and generate

$$Descinf([A, val_A] \wedge [DEC, val_{DEC}])$$
$$=Descinf([A, val_A]) \cap Descinf([DEC, val_{DEC}]),$$

$Descsup([A, val_A] \wedge [DEC, val_{DEC}])$
$=Descsup([A, val_A]) \cap Descsup([DEC, val_{DEC}])$.
If $[A, val_A] \wedge [DEC, val_{DEC}]$ satisfies either ($CASE$ A) or ($CASE$ B) in Algorithm 1, this descriptor is added to $CANDIDATE(2)$. Furthermore, we examine $minacc([A, val_A] \wedge [DEC, val_{DEC}])$ in (Proc 2-2) according to Proposition 4. The complexity of (STEP 2) depends upon the number of combined descriptors $[A, val_A] \wedge [DEC, val_{DEC}]$.

(STEP 3) (Repetition of STEP 2 on $CANDIDATE(n)$)
For each $DESC_1$ and $DESC_2$ in $CANDIDATE(n)$, we generate a conjunction $DESC_1 \wedge DESC_2$. For such conjunctions, we apply the same procedure as (STEP 2).

In the execution, two sets $Descinf([CON, \zeta])$ and $Descsup([CON, \zeta])$ are stored in arrays, and we can obtain $Descinf([CON, \zeta] \wedge [DEC, \eta])$ by using the intersection operation $Descinf([CON, \zeta]) \cap Descinf([DEC, \eta])$. The same property holds for $Descsup([CON, \zeta] \wedge [DEC, \eta])$. Therefore, it is easy to obtain $CANDIDATE(n+1)$ from $CANDIDATE(n)$. This is a merit of employing equivalence classes, and this is the characteristics of rough set theory. In *Apriori* algorithm, such $Descinf$ and $Descsup([CON, \zeta])$ are not employed, and the total search of a database is executed for generating every combination of descriptors. It will be necessary to consider the merit and demerit of handling two sets $Descinf([CON, \zeta])$ and $Descsup([CON, \zeta])$ in the next research.

Apriori algorithm employs an equivalence class for each descriptors, and handles only deterministic information. On the other hand, Algorithm 1 employs the minimum and the maximum sets of an equivalence class, i.e., $Descinf$ and $Descsup$, and handles non-deterministic information as well as deterministic information. In Algorithm 1, it takes twice steps of *Apriori* algorithm for manipulating equivalence classes. The rest is almost the same as *Apriori* algorithm, therefore the complexity of Algorithm 1 will be almost the same as *Apriori* algorithm.

6 Concluding Remarks and Future Work

We proposed rule generation based on *lower approximation strategy* and *upper approximation strategy* in $NISs$. We employed $Descinf$, $Descsup$ and the concept of large item set in *Apriori* algorithm, and proposed two extended *Apriori* algorithms in $NISs$. These extended algorithms do not depend upon the number of derived $DISs$, and the complexity of these extended algorithms is almost the same as *Apriori* algorithm. We implemented the extended algorithms, and applied them to some data sets. According to these utility programs, we can explicitly handle not only deterministic information but also non-deterministic information.

Now, we briefly show the application to Hepatitis data in UCI Machine Learning Repository [37]. In reality, we applied our programs to Hepatitis data. This data consists of 155 objects, 20 attributes. There are 167 missing values, which are about 5.4% of total data. The number of objects without missing values is 80, namely the number is about the half of total data. In usual analyzing tools, it may be difficult to handle total 155 objects.

We employ a list for expressing non-deterministic information, for example, [red,green], [red,blue] for $\{red, green\}$ and $\{red, blue\}$ in Table 2. This syntax is so simple that we can easily generate data of $NISs$ by using Excel. As for Hepatitis data, we loaded this data into Excel, and replaced each missing value (? symbol) with a list of all possible attribute values. For some numerical values, the discretized attribute values are also given in the data set. For example, in the 15th attribute BILIRUBIN, attribute values are discretized to the six attribute values, i.e., 0.39, 0.80, 1.20, 2.00, 3.00, 4.00. We employed these discretized values in some attributes. The following is a part of the real revised Hepatitis data in Excel. There are 78732 ($=2 \times 6 \times 9^4$) derived $DISs$ for these six objects. Probably, it seems hard to handle all derived $DISs$ for total 155 objects sequentially.

```
155     //Number of objects
20      //Number of Attributes
2 30 2 1 2 2 2 2 1 2 2 2 2 2 0.8 80 13 3.8 [10,20,30,40,50,60,70,80,90] 1
2 50 1 1 2 1 2 2 1 2 2 2 2 2 0.8 120 13 3.8 [10,20,30,40,50,60,70,80,90] 1
2 70 1 2 2 1 2 2 2 2 2 2 2 2 0.8 80 13 3.8 [10,20,30,40,50,60,70,80,90] 1
2 30 1 [1,2] 1 2 2 2 2 2 2 2 2 2 0.8 33 13 3.8 80 1
2 30 1 2 2 2 2 2 2 2 2 2 2 2 0.8 [33,80,120,160,200,250] 200 3.8 -
        [10,20,30,40,50,60,70,80,90] 1
2 30 1 2 2 2 2 2 2 2 2 2 2 2 0.8 80 13 3.8 70 1
                :           :           :
```

The decision attribute is the first attribute CLASS (1:die, 2:live), and we fixed α=0.25 and β=0.85. Let us show the results of two cases.

(CASE 1) Obtained Rules from 80 Objects without Missing Values
It is possible to apply our programs to the standard $DISs$. For 80 objects, it took 0.015(sec), and 14 rules including the following are generated.

```
[AGE,30]=>[Class,live] (support=0.287,accuracy=0.958),
[ASCITES,yes]=>[CLASS,live] (support=0.775,accuracy=0.912),
[ALBUMIN,4.5]=>[CLASS,live] (support=0.287,accuracy=0.958).
```

(CASE 2) Obtained Rules from 155 Objects with 167 Missing Values
Due to two strategies, 22 rules and 25 rules are generated, respectively. It took 0.064(sec). Let us show every rule, which is obtained by *upper approximation strategy* but is not obtained by *lower approximation strategy*. Namely, every rule is in boundary set $Rule(0.25, 0.85, UA) - Rule(0.25, 0.85, LA)$. There are three such rules.

```
[Alk_PHOSPHATE,80]=>[CLASS,live]
 (minsupp=0.25,minacc=0.841,maxsupp=0.348,maxacc=0.857)
[ANOREXIA,yes]&[SGOT,13]=>[CLASS,live]
 (minsupp=0.25,minacc=0.829,maxsupp=0.381,maxacc=0.855)
```

```
[SPLEEN_PALPABLE,yes]&[SGOT,13]=>[CLASS,live]
  (minsupp=0.25,minacc=0.848,maxsupp=0.368,maxacc=0.877)
```

In the 17th attribute SGOT, there are four missing values. The above two rules with descriptor [SGOT,13] depend upon these four missing values. These rules show us the difference between *lower approximation strategy* and *upper approximation strategy*.

We are also focusing on the difference between rule generation in $DISs$ and $NISs$. Let us suppose a NIS. We remove every object with non-deterministic information from the NIS, and we obtain a DIS. We are interested in rules, which are not obtained from the DIS but obtained from the NIS.

According to some experiments including Hepatitis data and Mammographic data in UCI repository, we verified our utility programs work well, even if there are huge number of derived $DISs$. However, we have not analyzed the meaning of the obtained rules. Because, the main issue of this paper is to establish the framework and to implement algorithms. From now on, we will apply our utility programs to real data with missing values, and we want to obtain meaningful rules from $NISs$. Our research is not toward rule generation from data with a large number of objects, but it is toward rule generation from incomplete data with a large number of derived $DISs$.

This paper is a revised and extended version of papers [31, 32].

Acknowledgment. The authors would be grateful to anonymous referees for their useful comments. This work is partly supported by the Grant-in-Aid for Scientific Research (C) (No.16500176, No.18500214), Japan Society for the Promotion of Science.

References

1. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: Proceedings of the 20th Very Large Data Base, pp. 487–499 (1994)
2. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.: Fast Discovery of Association Rules. In: Advances in Knowledge Discovery and Data Mining, pp. 307–328. AAAI/MIT Press (1996)
3. Demri, S., Orłowska, E.: Incomplete Information: Structure, Inference, Complexity. Monographs in Theoretical Computer Science. Springer, Heidelberg (2002)
4. Grzymala-Busse, J.: On the Unknown Attribute Values in Learning from Examples. In: Raś, Z.W., Zemankova, M. (eds.) ISMIS 1991. LNCS (LNAI), vol. 542, pp. 368–377. Springer, Heidelberg (1991)
5. Grzymala-Busse, J.: A New Version of the Rule Induction System LERS. Fundamenta Informaticae 31, 27–39 (1997)
6. Grzymala-Busse, J., Werbrouck, P.: On the Best Search Method in the LEM1 and LEM2 Algorithms. Incomplete Information: Rough Set Analysis 13, 75–91 (1998)
7. Grzymala-Busse, J.: Data with Missing Attribute Values: Generalization of Indiscernibility Relation and Rule Induction. Transactions on Rough Sets 1, 78–95 (2004)

8. Grzymala-Busse, J.: Incomplete data and generalization of indiscernibility relation, definability, and approximations. In: Śezak, D., Wang, G., Szczuka, M.S., Düntsch, I., Yao, Y. (eds.) RSFDGrC 2005. LNCS (LNAI), vol. 3641, pp. 244–253. Springer, Heidelberg (2005)
9. Komorowski, J., Pawlak, Z., Polkowski, L., Skowron, A.: Rough Sets: a tutorial. In: Pal, S., Skowron, A. (eds.) Rough Fuzzy Hybridization, pp. 3–98. Springer, Heidelberg (1999)
10. Kryszkiewicz, M.: Rules in Incomplete Information Systems. Information Sciences 113, 271–292 (1999)
11. Kryszkiewicz, M., Rybinski, H.: Computation of Reducts of Composed Information Systems. Fundamenta Informaticae 27, 183–195 (1996)
12. Kryszkiewicz, M.: Maintenance of Reducts in the Variable Precision Rough Sets Model. ICS Research Report 31/94, Warsaw University of Technology (1994)
13. Lipski, W.: On Semantic Issues Connected with Incomplete Information Data Base. ACM Trans. DBS 4, 269–296 (1979)
14. Lipski, W.: On Databases with Incomplete Information. Journal of the ACM 28, 41–70 (1981)
15. Nakamura, A., Tsumoto, S., Tanaka, H., Kobayashi, S.: Rough Set Theory and Its Applications. Journal of Japanese Society for AI 11, 209–215 (1996)
16. Nakamura, A.: A Rough Logic based on Incomplete Information and Its Application. International Journal of Approximate Reasoning 15, 367–378 (1996)
17. Nakata, M., Sakai, H.: Rough-set-based Approaches to Data Containing Incomplete Information: Possibility-based Cases. In: Nakamatsu, K., Abe, J. (eds.) Advances in Logic Based Intelligent Systems. Frontiers in Artificial Intelligence and Applications, vol. 132, pp. 234–241. IOS Press, Amsterdam (2005)
18. Nakata, M., Sakai, H.: Lower and Upper Approximations in Data Tables Containing Possibilistic Information. Transactions on Rough Sets 7, 170–189 (2007)
19. Orłowska, E.: What You Always Wanted to Know about Rough Sets. In: Incomplete Information: Rough Set Analysis, vol. 13, pp. 1–20. Physica-Verlag (1998)
20. Orłowska, E., Pawlak, Z.: Representation of Nondeterministic Information. Theoretical Computer Science 29, 27–39 (1984)
21. Pawlak, Z.: Rough Sets. Kluwer Academic Publisher, Dordrecht (1991)
22. Pawlak, Z.: Some Issues on Rough Sets. Transactions on Rough Sets 1, 1–58 (2004)
23. Polkowski, L., Skowron, A. (eds.): Rough Sets in Knowledge Discovery 1. Studies in Fuzziness and Soft Computing, vol. 18. Physica-Verlag (1998)
24. Polkowski, L., Skowron, A. (eds.): Rough Sets in Knowledge Discovery 2. Studies in Fuzziness and Soft Computing, vol. 19. Physica-Verlag (1998)
25. Rough Set Software. Bulletin of Int'l. Rough Set Society 2, 15–46 (1998)
26. Sakai, H.: Effective Procedures for Handling Possible Equivalence Relations in Non-deterministic Information Systems. Fundamenta Informaticae 48, 343–362 (2001)
27. Sakai, H.: Effective Procedures for Data Dependencies in Information Systems. In: Rough Set Theory and Granular Computing. Studies in Fuzziness and Soft Computing, vol. 125, pp. 167–176. Springer, Heidelberg (2003)
28. Sakai, H., Okuma, A.: Basic Algorithms and Tools for Rough Non-deterministic Information Analysis. In: Peters, J.F., Skowron, A., Grzymała-Busse, J.W., Kostek, B.z., Świniarski, R.W., Szczuka, M.S. (eds.) Transactions on Rough Sets I. LNCS, vol. 3100, pp. 209–231. Springer, Heidelberg (2004)
29. Sakai, H., Nakata, M.: An Application of Discernibility Functions to Generating Minimal Rules in Non-deterministic Information Systems. Journal of Advanced Computational Intelligence and Intelligent Informatics 10, 695–702 (2006)

30. Sakai, H.: On a Rough Sets Based Data Mining Tool in Prolog: An Overview. In: Umeda, M., Wolf, A., Bartenstein, O., Geske, U., Seipel, D., Takata, O. (eds.) INAP 2005. LNCS (LNAI), vol. 4369, pp. 48–65. Springer, Heidelberg (2006)
31. Sakai, H., Nakata, M.: On Possible Rules and Apriori Algorithm in Non-deterministic Information Systems. In: Greco, S., Hata, Y., Hirano, S., Inuiguchi, M., Miyamoto, S., Nguyen, H.S., Słowiński, R. (eds.) RSCTC 2006. LNCS (LNAI), vol. 4259, pp. 264–273. Springer, Heidelberg (2006)
32. Sakai, H., Ishibashi, R., Koba, K., Nakata, M.: On Possible Rules and Apriori Algorithm in Non-deterministic Information Systems 2. In: An, A., Stefanowski, J., Ramanna, S., Butz, C.J., Pedrycz, W., Wang, G. (eds.) RSFDGrC 2007. LNCS (LNAI), vol. 4482, pp. 280–288. Springer, Heidelberg (2007)
33. Skowron, A., Rauszer, C.: The Discernibility Matrices and Functions in Information Systems. In: Intelligent Decision Support - Handbook of Advances and Applications of the Rough Set Theory, pp. 331–362. Kluwer Academic Publishers, Dordrecht (1992)
34. Stefanowski, J., Tsoukias, A.: On the Extension of Rough Sets under Incomplete Information. In: Zhong, N., Skowron, A., Ohsuga, S. (eds.) RSFDGrC 1999. LNCS (LNAI), vol. 1711, pp. 73–81. Springer, Heidelberg (1999)
35. Stefanowski, J., Tsoukias, A.: Incomplete Information Tables and Rough Classification. Computational Intelligence 7, 212–219 (2001)
36. Tsumoto, S.: Knowledge Discovery in Clinical Databases and Evaluation of Discovered Knowledge in Outpatient Clinic. Information Sciences 124, 125–137 (2000)
37. UCI Machine Learning Repository, `http://mlearn.ics.uci.edu/MLRepository.html`
38. Ziarko, W.: Variable Precision Rough Set Model. Journal of Computer and System Sciences 46, 39–59 (1993)

On Extension of Dependency and Consistency Degrees of Two Knowledges Represented by Covering

P. Samanta[1] and Mihir K. Chakraborty[2]

[1] Department of Mathematics, Katwa College
Katwa, Burdwan, West Bengal, India
pulak_samanta06@yahoo.co.in

[2] Department of Pure Mathematics, University of Calcutta
35, Ballygunge Circular Road, Kolkata-700019, India
mihirc99@vsnl.com

Abstract. Knowledge of an agent depends on the granulation procedure adopted by the agent. The knowledge granules may form a partition of the universe or a covering. In this paper dependency degrees of two knowledges have been considered in both the cases. A measure of consistency and inconsistency of knowledges are also discussed. This paper is a continuation of our earlier work [3].

Keywords: Rough sets, elementary category(partition, covering of knowledge), dependency degree, consistency degree.

1 Introduction

Novotný and Pawlak defined a dependency degree between two knowledges given by two partitions on a set [6,7,8,9]. Knowledge is given by indiscernibility relations on the universe and indiscernibility relation is taken to be an equivalence relation. But in many situations the indiscernibility relation fails to be transitive. Hence the clusters or granules of knowledge overlap. This observation gives rise to the study of Rough Set Theory based on coverings instead of partitions [2,10,11,13,14,15,16].

In [3] the present authors introduced the notions of consistency degree and inconsistency degree of two knowledges given by partitions of the universe using the dependency degree defined by Novotný and Pawlak. In this paper some more investigations in that direction have been carried out but the main emphasis is laid on defining the dependency degree of two knowledges when they are given by coverings in general, not by partitions only. Now, in the covering based approximation systems lower and upper approximations of a set are defined in at least five different ways [10]. All of these approximations reduce to the standard Pawlakian approximations when the underlying indiscernibility relation turns out to be equivalence. We have in this paper used four of them of which one is the classical one. As a consequence, four different dependency degrees arise.

It is interestingly observed that the properties of partial dependency that were developed in [3,6,9] hold good in the general case of covering based approximation system. The main results on covering are placed in section 3. Depending upon this generalized notion of dependency, consistency degree and inconsistency degree between two such knowledges have been defined.

2 Dependency of Knowledge Based on Partition

We would accept the basic philosophy that a knowledge of an agent about an universe is her ability to categorize objects inhabiting it through information received from various sources or perception in the form of attribute-value data. For this section we start with the indiscernibility relation caused by the attribute-value system. So, knowledge is defined as follows.

Definition 1. *Knowledge : A knowledge is a pair, $< U, P >$ where U is a non-empty finite set and P is an equivalence relation on U. P will also denote the partition generated by the equivalence relation.*

Definition 2. *Finer and Coarser Knowledge : A knowledge P is said to be finer than the knowledge Q if every block of the partition P is included in some block of the partition Q. In such a case Q is said to coarser than P. We shall write it as $P \preceq Q$.*

We recall a few notions due to Pawlak (and others) e.g P-positive region of Q and based upon it dependency-degree of knowledges.

Definition 3. *Let P and Q be two equivalence relations over U. The P-positive region of Q, denoted by $Pos_P(Q)$ is defined by $Pos_P(Q) = \bigcup_{X \in U/Q} \underline{P}X$, where $\underline{P}X = \{\bigcup Y \in U/P : Y \subseteq X\}$ called P-lower approximation of X.*

Definition 4. *Dependency degree : Knowledge Q depends in a degree k $(0 \leq k \leq 1)$ on knowledge P , written as $P \Rightarrow_k Q$, iff $k = \frac{CardPos_P(Q)}{CardU}$ where card denotes cardinality of the set.*

If $k = 1$, we say that Q totally depends on P and we write $P \Rightarrow Q$; and if $k = 0$ we say that Q is totally independent of P.

Viewing from the angle of multi-valuedness one can say that the sentence 'The knowledge Q depends on the knowledge P' instead of being only 'true'(1) or 'false'(0) may receive other intermediate truth-values, the value k being determined as above. This approach justifies the term 'partial dependency' as well.

In propositions 1,2 and 3, we enlist some elementary, often trivial, properties of dependency degree some of them being newly exercised but most of which are present in [6,9]. Some of these properties e.g. proposition 3(v) will constitute the basis of definitions and results of the next section.

Proposition 1
(i) $[x]_{P_1 \cap P_2} = [x]_{P_1} \cap [x]_{P_2}$,
(ii) If $P \Rightarrow Q$ *and* $R \preceq P$ *then* $R \Rightarrow Q$,
(iii)If $P \Rightarrow Q$ *and* $Q \preceq R$ *then* $P \Rightarrow R$,
(iv)If $P \Rightarrow Q$ *and* $Q \Rightarrow R$ *then* $P \Rightarrow R$,
(v)If $P \Rightarrow R$ *and* $Q \Rightarrow R$ *then* $P \cap Q \Rightarrow R$,
(vi) If $P \Rightarrow R \cap Q$ *then* $P \Rightarrow R$ *and* $P \Rightarrow Q$,
(vii) If $P \Rightarrow Q$ *and* $Q \cap R \Rightarrow T$ *then* $P \cap R \Rightarrow T$,
(viii) If $P \Rightarrow Q$ *and* $R \Rightarrow T$ *then* $P \cap R \Rightarrow Q \cap T$.

Proposition 2
(i) If $P' \preceq P$ *then* $\underline{P'}X \supseteq \underline{P}X$,
(ii) If $P \Rightarrow_a Q$ *and* $P' \preceq P$ *then* $P' \Rightarrow_b Q$ *where* $b \geq a$,
(iii) If $P \Rightarrow_a Q$ *and* $P \preceq P'$ *then* $P' \Rightarrow_b Q$ *where* $b \leq a$,
(iv) If $P \Rightarrow_a Q$ *and* $Q' \preceq Q$ *then* $P \Rightarrow_b Q'$ *where* $b \leq a$,
(v) If $P \Rightarrow_a Q$ *and* $Q \preceq Q'$ *then* $P \Rightarrow_b Q'$ *where* $a \leq b$.

Proposition 3
(i) If $R \Rightarrow_a P$ *and* $Q \Rightarrow_b P$ *then* $R \cap Q \Rightarrow_c P$
for some $c \geq Max(a, b)$,
(ii) If $R \cap P \Rightarrow_a Q$ *then* $R \Rightarrow_b Q$ *and* $P \Rightarrow_c Q$ *for some* $b, c \leq a$,
(iii) If $R \Rightarrow_a Q$ *and* $R \Rightarrow_b P$ *then* $R \Rightarrow_c Q \cap P$ *for some* $c \leq Min(a, b)$,
(iv) If $R \Rightarrow_a Q \cap P$ *then* $R \Rightarrow_b Q$ *and* $R \Rightarrow_c P$ *for some* $b, c \geq a$,
(v) If $R \Rightarrow_a P$ *and* $P \Rightarrow_b Q$ *then* $R \Rightarrow_c Q$ *for some* $c \geq a + b - 1$.

3 Dependency of Knowledge Based on Covering

A covering $\mathcal{C}$ of a set U is a collection of subsets $\{C_i\}$ of U such that $\cup C_i = U$. It is often important to define a knowledge in terms of covering and not by partition which is a special case of covering. Given a covering $\mathcal{C}$ one can define a binary relation $R^{\mathcal{C}}$ on U which is a tolerance relation (reflexive, symmetric) by $xR^{\mathcal{C}}y$ holds iff $x, y \in C_i$ for some i, where the set $\{C_i\}$ constitute the covering.

Definition 5. *A tolerance space is a structure* $S = < U, R >$, *where* U *is a nonempty set of objects and* R *is a reflexive and symmetric binary relation defined on* U.

A tolerance class of a tolerance space $< U, R >$ is a maximal subset of U such that any two elements of it are mutually related.

In the context of knowledge when the indiscernibility relation R is only reflexive and symmetric (and not necessarily transitive) the approximation system $< U, R >$ is a tolerance space. In such a case the granules of the Knowledge may be formed in many different ways. Since the granules are not necessarily disjoint it is worthwhile to talk about granulation around an object $x \in U$. Now the most natural granule at x is the set $\{y : xRy\}$. This set is generally denoted by R_x. But any element C_i of the covering $\mathcal{C}$ can also be taken as a granule around x where $x \in C_i$. There may be others. So, depending upon various ways of perceiving a granule, various definitions of lower approximations (and hence the upper approximations as their duals) of a set may be given. We shall consider them below. Now any covering gives rise to a unique partition. By $\mathcal{P}$ we denote the partition corresponding to the covering $\mathcal{C}$.

Definition 6. *[1,2] A covering is said to be genuine covering if $C_i \subseteq C_j$ implies $C_i = C_j$.*

For any genuine covering $\mathcal{C}$ it is immediate that the elements of $\mathcal{C}$ are all tolerance classes of the relation $R^{\mathcal{C}}$.

Definition 7. *Let two finite coverings $\mathcal{C}_1$ and $\mathcal{C}_2$ be given by $\mathcal{C}_1 = \{C_1, C_2, ...C_n\}$ and $\mathcal{C}_2 = \{C'_1, C'_2, ...C'_m\}$. Then $\mathcal{C}_1 \cap \mathcal{C}_2$ is the collection $\{C_i \cap C'_j$ where $i = 1, 2, ...n; j = 1, 2, ...m\}$.*

Example 1. Let $\mathcal{C}_1 = \{\{1, 2, 3\}, \{2, 3, 4\}, \{5, 6, 7\}, \{6, 7, 8\}\}$ and $\mathcal{C}_2 = \{\{1, 2, 3, 4\}, \{3, 4, 5, 6\}, \{5, 6, 7, 8\}\}$.

Then $\mathcal{C}_1 \cap \mathcal{C}_2 = \{\{1, 2, 3\}, \{3\}, \{2, 3, 4\}, \{3, 4\}, \{5, 6\}, \{5, 6, 7\}, \{6\}, \{6, 7, 8\}\}$.

Definition 8. *We shall say that a covering $\mathcal{C}_1$ is finer than a covering $\mathcal{C}_2$ written as $\mathcal{C}_1 \preceq \mathcal{C}_2$ iff $\forall C'_j \in \mathcal{C}_2\ \exists\ C_{j1}, C_{j2}, ..., C_{jn}$ such that $C'_j = C_{j1} \cup C_{j2} \cup ... \cup C_{jn}$ where, $C_{j1}, C_{j2}, ..., C_{jn} \in \mathcal{C}_1$ i.e. every element of $\mathcal{C}_2$ may be expressed as the union of some elements of $\mathcal{C}_1$.*

Let R be a tolerance relation in U. Then the family $\mathcal{C}(R)$ of all tolerances classes of R is a covering of U. The pair $(U, \mathcal{C})$ will be called generalized approximation space, where U is a set and $\mathcal{C}$ is a covering of U. We shall however assume U to be finite in the sequel.

Let $(U, \mathcal{C})$ be a generalized approximation space and $\mathcal{C} = \{C_1, C_2, ...C_n\}$. The indiscernibility neighborhood of an element $x \in U$ is the set $N^{\mathcal{C}}_x = \bigcup\{C_i : x \in C_i\}$. In fact $N^{\mathcal{C}}_x$ is the same as $R^{\mathcal{C}}_x$.

For any $x \in U$ the set $P^{\mathcal{C}}_x = \{y \in U : \forall C_i(x \in C_i \Leftrightarrow y \in C_i)\}$ will be called *kernel of x*. Let $\mathcal{P}$ be the family of all kernels $(U, \mathcal{C})$ i.e. $\mathcal{P} = \{P^{\mathcal{C}}_x : x \in U\}$. Clearly $\mathcal{P}$ is a partition of U.

Definition 9. *[10] Let X be a subset of U. Then the lower and upper approximations are defined as follows :*

$$\underline{\mathcal{C}}_1(X) = \{x : N^{\mathcal{C}}_x \subseteq X\}$$
$$\overline{\mathcal{C}}^1(X) = \bigcup\{C_i : C_i \cap X \neq \phi\}$$

$$\underline{\mathcal{C}}_2(X) = \bigcup\{N^{\mathcal{C}}_x : N^{\mathcal{C}}_x \subseteq X\}$$
$$\overline{\mathcal{C}}^2(X) = \{z : \forall y(z \in N^{\mathcal{C}}_x \Rightarrow N^{\mathcal{C}}_x \cap X \neq \phi)\}$$

$$\underline{\mathcal{C}}_3(X) = \bigcup\{C_i, C_i \subseteq X \ for\ some\ C_i \in \mathcal{C}_1\}$$
$$\overline{\mathcal{C}}^3(X) = \{y : \forall C_i(y \in C_i \Rightarrow C_i \cap X \neq \phi)\}$$

$\underline{\mathcal{C}}_4(X) = \bigcup\{P_x^{\mathcal{C}} : P_x^{\mathcal{C}} \subseteq X\}$
$\overline{\mathcal{C}}^4(X) = \bigcup\{\{P_x^{\mathcal{C}} : \{P_x^{\mathcal{C}} \cap X \neq \phi\}$

Proposition 4. *If $\mathcal{C}_1 \preceq \mathcal{C}_2$ then $P_1 \preceq P_2$ where P_1, P_2 are the partitions corresponding to $\mathcal{C}_1$, and $\mathcal{C}_2$ respectively.*

Proposition 5. *If $\mathcal{C}_1 \preceq \mathcal{C}_2$ then for any $X \subseteq U$, $\underline{\mathcal{C}_1}_i(X) \supseteq \underline{\mathcal{C}_2}_i(X)$ and $\overline{\mathcal{C}_1}^i(X) \subseteq \overline{\mathcal{C}_2}^i(X)$ for $i = 1, 2, 3, 4$.*

Example 2. Let $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and $\mathcal{C} = \{\{1, 2\}, \{1, 2, 3\}, \{4, 6\}, \{6, 7, 9\}, \{8, 9\}, \{5, 10\}\}$. Let $A = \{1, 2, 4, 6, 9, 10\}$. Then $\underline{\mathcal{C}}_1(A) = \{4\}$, $\underline{\mathcal{C}}_2(A) = \{4, 6\}$, $\underline{\mathcal{C}}_3(A) = \{1, 2, 4, 6\}$, $\underline{\mathcal{C}}_4(A) = \{1, 2, 4, 6, 9\}$.

Let $B = \{3, 9, 10\}$. Then $\overline{\mathcal{C}}^1(B) = \{1, 2, 3, 5, 6, 7, 8, 9, 10\}$, $\overline{\mathcal{C}}^2(B) = \{1, 2, 3, 5, 7, 8, 9, 10\}$, $\overline{\mathcal{C}}^3(B) = \{3, 5, 7, 8, 9, 10\}$, $\overline{\mathcal{C}}^4(B) = \{3, 9\}$

Proposition 6. *Propositions* 1, 2, 3 *except* 3(*v*) *of section* 2 *also hold in this generalized case.*

Definition 10. *We define $\mathcal{C}_1$-Positive region of $\mathcal{C}_2$ as $Pos_{\mathcal{C}_1}\mathcal{C}_2 = \bigcup_{X\in\mathcal{C}_2} \underline{\mathcal{C}_1}(X)$.*

Definition 11. *Dependency degree with respect to covering : $\mathcal{C}_1$ depends in a degree k $(0 \leq k \leq 1)$ on $\mathcal{C}_2$, written as $\mathcal{C}_1 \Rightarrow_k \mathcal{C}_2$, iff $k = \frac{|Pos_{\mathcal{C}_1}\mathcal{C}_2|}{|U|}$ where $|X|$ denotes cardinality of the set X. We shall also write $k = Dep(\mathcal{C}_1, \mathcal{C}_2)$. If $k = 1$, $\mathcal{C}_1$ is said to be totally dependent on $\mathcal{C}_2$ and we write $\mathcal{C}_1 \Rightarrow \mathcal{C}_2$; and if $k = 0$ we say that $\mathcal{C}_2$ is totally independent of $\mathcal{C}_1$.*

Since we have four kinds of lower approximations, we have, four different $\mathcal{C}_1$-Positive region of $\mathcal{C}_2$ viz. $Pos^i_{\mathcal{C}_1}\mathcal{C}_2$ with respect to $\underline{\mathcal{C}}_i(X)$ for $i = 1, 2, 3, 4$ and also four different kinds of Dependencies viz. $Dep^i(\mathcal{C}_1, \mathcal{C}_2)$ for $i = 1, 2, 3, 4$.

Clearly, $\bigcup_{X\in\mathcal{C}_2}\{x : N_x^{\mathcal{C}_1} \subseteq X\} \subseteq \bigcup_{X\in\mathcal{C}_2}\{N_x^{\mathcal{C}_1} : x \in \mathcal{C}_1, N_x^{\mathcal{C}_1} \subseteq X\} \subseteq \bigcup_{X\in\mathcal{C}_2}\{C_i,$ $C_i \subseteq X \ for\ some\ C_i \in \mathcal{C}_1\} \subseteq \bigcup_{X\in\mathcal{C}_2}\{P_x^{\mathcal{C}} : P_x^{\mathcal{C}} \subseteq X\}$.

This implies, $Pos^1_{\mathcal{C}_1}\mathcal{C}_2 \subseteq Pos^2_{\mathcal{C}_1}\mathcal{C}_2 \subseteq Pos^3_{\mathcal{C}_1}\mathcal{C}_2 \subseteq Pos^4_{\mathcal{C}_1}\mathcal{C}_2$. So, we have, $\frac{|Pos^1_{\mathcal{C}_1}\mathcal{C}_2|}{|U|} \leq \frac{|Pos^2_{\mathcal{C}_1}\mathcal{C}_2|}{|U|} \leq \frac{|Pos^3_{\mathcal{C}_1}\mathcal{C}_2|}{|U|} \leq \frac{|Pos^4_{\mathcal{C}_1}\mathcal{C}_2|}{|U|}$.

So, the following proposition is obtained.

Proposition 7. $Dep^1(\mathcal{C}_1, \mathcal{C}_2) \leq Dep^2(\mathcal{C}_1, \mathcal{C}_2) \leq Dep^3(\mathcal{C}_1, \mathcal{C}_2) \leq Dep^4(\mathcal{C}_1, \mathcal{C}_2)$.

Example 3. Consider $\mathcal{C}_1 = \{\{1, 2, 3\}, \{2, 3, 4\}, \{5, 6, 7\}, \{6, 7, 8\}\}$ and $\mathcal{C}_2 = \{\{1, 2, 3, 4\}, \{3, 4, 5, 6\}, \{5, 6, 7, 8\}\}$.

Then $Dep^1(\mathcal{C}_1, \mathcal{C}_2) = 1$, $Dep^2(\mathcal{C}_1, \mathcal{C}_2) = 1$, $Dep^3(\mathcal{C}_1, \mathcal{C}_2) = 1$, $Dep^4(\mathcal{C}_1, \mathcal{C}_2) = 1$.
Also, $Dep^1(\mathcal{C}_2, \mathcal{C}_1) = 0$, $Dep^2(\mathcal{C}_2, \mathcal{C}_1) = 0$, $Dep^3(\mathcal{C}_2, \mathcal{C}_1) = 0$, $Dep^4(\mathcal{C}_2, \mathcal{C}_1) = 1$.

Example 4. Let us Consider $\mathcal{C}_1 = \{\{1, 2, 3\}, \{3, 4, 8\}, \{6, 7, 8\}, \{8, 9\}\}$ and $\mathcal{C}_2 = \{\{1, 2, 3, 4\}, \{5, 8\}, \{6, 7\}, \{8, 9\}\}$.

Then $Dep^1(\mathcal{C}_1, \mathcal{C}_2) = \frac{1}{3}$, $Dep^2(\mathcal{C}_1, \mathcal{C}_2) = \frac{1}{3}$, $Dep^3(\mathcal{C}_1, \mathcal{C}_2) = \frac{5}{9}$, $Dep^4(\mathcal{C}_1, \mathcal{C}_2) = 1$.
Also, $Dep^1(\mathcal{C}_2, \mathcal{C}_1) = \frac{1}{3}$, $Dep^2(\mathcal{C}_2, \mathcal{C}_1) = \frac{1}{3}$, $Dep^3(\mathcal{C}_2, \mathcal{C}_1) = \frac{4}{9}$, $Dep^4(\mathcal{C}_2, \mathcal{C}_1) = \frac{5}{9}$.

Observation

(i) $\mathcal{C}_1 \Rightarrow \mathcal{C}_2$
iff $Pos_{\mathcal{C}_1}\mathcal{C}_2 = U$
iff $\bigcup_{X\in\mathcal{C}_2} \underline{\mathcal{C}_1}_1(X) = U$
iff $\bigcup_{X\in\mathcal{C}_2} \{x : N_x^{\mathcal{C}} \subseteq X\} = U$
iff $\forall x \in U, N_x^{\mathcal{C}_1} \subseteq X$ for some $X \in \mathcal{C}_2$.

Also $\mathcal{C}_1 \Rightarrow_0 \mathcal{C}_2$
iff $Pos_{\mathcal{C}_1}\mathcal{C}_2 = \phi$
iff $\bigcup_{X\in\mathcal{C}_2} \underline{\mathcal{C}_1}_1(X) = \phi$
iff $\bigcup_{X\in\mathcal{C}_2} \{x : N_x^{\mathcal{C}_1} \subseteq X\} = \phi$
iff $\forall x \in U$, there does not exists any $X \in \mathcal{C}_2$ such that $N_x^{\mathcal{C}_1} \subseteq X$.

(ii) $\mathcal{C}_1 \Rightarrow \mathcal{C}_2$
iff $Pos_{\mathcal{C}_1}\mathcal{C}_2 = U$
iff $\bigcup_{X\in\mathcal{C}_2} \underline{\mathcal{C}_1}_2(X) = U$
iff $\bigcup_{X\in\mathcal{C}_2} \bigcup_x \{N_x^{\mathcal{C}_1} : N_x^{\mathcal{C}_1} \subseteq X\} = U$
iff $\forall x \in U, N_x^{\mathcal{C}_1} \subseteq X$ for some $X \in \mathcal{C}_2$.

Also $\mathcal{C}_1 \Rightarrow_0 \mathcal{C}_2$
iff $Pos_{\mathcal{C}_1}\mathcal{C}_2 = \phi$
iff $\bigcup_{X\in\mathcal{C}_2} \underline{\mathcal{C}_1}_2(X) = \phi$
iff $\bigcup_{X\in\mathcal{C}_2} \bigcup_x \{N_x^{\mathcal{C}_1} : N_x^{\mathcal{C}_1} \subseteq X\} = \phi$
iff $\forall x \in U$, there does not exists any $X \in \mathcal{C}_2$ such that $N_x^{\mathcal{C}_1} \subseteq X$.

(iii) $\mathcal{C}_1 \Rightarrow \mathcal{C}_2$
iff $Pos_{\mathcal{C}_1}\mathcal{C}_2 = U$
iff $\bigcup_{X\in\mathcal{C}_2} \underline{\mathcal{C}_1}_3(X) = U$
iff $\bigcup_{X\in\mathcal{C}_2} \bigcup_i \{C_i \in \mathcal{C}_1 : C_i \subseteq X\} = U$
iff each $C_i(\in \mathcal{C}_1) \subseteq X$, for some $X \in \mathcal{C}_2$.

Also $\mathcal{C}_1 \Rightarrow_0 \mathcal{C}_2$
iff $Pos_{\mathcal{C}_1}\mathcal{C}_2 = \phi$
iff $\bigcup_{X\in\mathcal{C}_2} \underline{\mathcal{C}_1}_3(X) = \phi$
iff $\bigcup_{X\in\mathcal{C}_2} \bigcup_i \{C_i \in \mathcal{C}_1 : C_i \subseteq X\} = \phi$
iff for any $C_i \in \mathcal{C}_1$ there does not exists any $X \in \mathcal{C}_2$ such that $C_i \subseteq X$.

(iv) $\mathcal{C}_1 \Rightarrow \mathcal{C}_2$
iff $Pos_{\mathcal{C}_1}\mathcal{C}_2 = U$
iff $\bigcup_{X\in\mathcal{C}_2} \underline{\mathcal{C}_1}_4(X) = U$

iff $\bigcup_{X\in\mathcal{C}_2}\bigcup_x\{P_x^{\mathcal{C}_1} : P_x^{\mathcal{C}_1}\subseteq X\}=U$
iff for all x, $P_x^{\mathcal{C}_1}\subseteq X$ for some $X\in\mathcal{C}_2$.

Also $\mathcal{C}_1\Rightarrow_0\mathcal{C}_2$
iff $Pos_{\mathcal{C}_1}\mathcal{C}_2=\phi$
iff $\bigcup_{X\in\mathcal{C}_2}\underline{\mathcal{C}_1}_4(X)=\phi$
iff $\bigcup_{X\in\mathcal{C}_2}\{P_x^{\mathcal{C}_1} : P_x^{\mathcal{C}_1}\subseteq X\}=\phi$
iff for all x, there does not exists any $X\in\mathcal{C}_2$ such that $P_x^{\mathcal{C}_1}\subseteq X$.

The sets $\mathcal{C}_1$ and $\mathcal{C}_2$ may be considered as two groups of classifying properties of the objects of the universe U. Properties belonging to any group may have overlapping extensions. Now if $\mathcal{C}_1\Rightarrow\mathcal{C}_2$ holds i.e. the dependency degree of $\mathcal{C}_2$ on $\mathcal{C}_1$ is 1 then the following is its characterization in the first two cases (i) and (ii): given any element x of the universe, the set of all objects satisfying at least one of the properties of x is included in the extension of at least one of the classifying properties belonging to the second group.

If, on the other hand $\mathcal{C}_1\Rightarrow_0\mathcal{C}_2$ holds, it follows that, $\forall x\in U$, $N_x^{\mathcal{C}_1}$ is not a subset of X for any $X\in\mathcal{C}_2$; that means for any element x there is at least one element y which shares at least one of the classificatory properties of the first group and does not have any of the classificatory properties belonging to the second group.

In the third case (iii) $\mathcal{C}_1\Rightarrow\mathcal{C}_2$ iff $\forall C_i\in\mathcal{C}_1$, $\exists C_j\in\mathcal{C}_2$ such that $C_i\subseteq C_j$ and $\mathcal{C}_1\Rightarrow_0\mathcal{C}_2$ iff $\forall C_i\in\mathcal{C}_1$ there does not exist any $C_j\in\mathcal{C}_2$ such that $C_i\subseteq C_j$.

The first condition means that the extension of any of the classificatory properties of the first group is a subset of the extension of at least one of the classificatory properties of the second. On the other hand the second one means : no classificatory property belonging to the first group implies any one of the classificatory property of the second group.

In the fourth case (iv) if x and y are equivalent with respect to the classificatory properties in the group $\mathcal{C}_1$ then x and y will share at least one of the classificatory properties with respect to $\mathcal{C}_2$ and vice versa.

4 Consistency of Knowledge Based on Partition

Two knowledges P and Q on U where P and Q are partitions may be considered as fully consistent if and only if $U/P=U/Q$, that is P,Q generate exactly the same granules. This is equivalent to $P\Rightarrow Q$ and $Q\Rightarrow P$. So, a natural measure of consistency degree of P and Q might be the truth-value of the non-classical sentence "Q depends on $P\wedge P$ depends on Q" computed by a suitable conjunction operator applied on the truth-values of the two component sentences Thus a binary predicate $Cons$ may be created such that $Cons(P,Q)$ will stand for the above conjunctive sentence. A triangular norm (or t-norm) used in fuzzy-literature and many-valued logic scenario is a potential candidate for computing $\wedge$. A t-norm is a mapping $t:[0,1]\to[0,1]$ satisfying (i) $t(a,1)=a$, (ii) $b\leq d$

implies $t(a,b) \leq t(a,d)$, (iii) $t(a,b) = t(b,a)$, (iv) $t(a,t(b,d)) = t(t(a,b),d)$. It follows that $t(a,0) = 0$. Typical examples of t-norm are :

$min(a,b)$ (Gödel),
$max(0, a+b-1)$ (Lukasicwicz),
$a \times b$ (Godo,Hajek).

These are conjunction operators used extensively and are in some sense the basic t-norms [4]. With $1-x$ as negation operator the De-Morgan dual of t-norms called s-norms are obtained as $s(a,b) = 1 - t(1-a, 1-b)$. Values of disjunctive sentences are computed by s-norms.

There is however a difficulty in using a t-norm in the present context. We would like to have the following assumptions to hold.

Assumption 1. Knowledges P,Q shall be fully consistent iff they generate the same partition.

Assumption 2. Knowledges P,Q shall be fully inconsistent iff no granule generated by one is contained in any granule generated by the other.

The translation of the above demands in mathematical terms is that the conjunction operator $\star$ should fulfill the conditions:

$\star(a,b) = 1$ iff $a = 1, b = 1$
and $\star(a,b) = 0$ iff $a = 0, b = 0$.

No t-norm satisfies the second. So we define consistency degree as follows:

Definition 12. *Let P and Q be two knowledges such that $P \Rightarrow_a Q$ and $Q \Rightarrow_b P$. The consistency degree between the two knowledges denoted by $Cons(P,Q)$ is given by $Cons(P,Q) = \frac{a+b+nab}{n+2}$, where n is a non negative integer.*

Definition 13. *Two knowledges P and Q are said to be fully consistent if $Cons(P,Q) = 1$.*

Two knowledge P and Q are said to be fully inconsistent if $Cons(P,Q) = 0$.

Example 5. (i) Let $U = \{1,2,3,4,5,6,7,8\}$ and the partitions be taken as $P = \{\{1,3,5\},\{2,4,6\},\{7,8\}\}$ and $Q = \{\{1,2,7\},\{3,4,8\},\{5,6\}\}$. Then $P \Rightarrow_0 Q$ and $Q \Rightarrow_0 P$. So, $Cons(P,Q) = 0$.
(ii) Let $U = \{1,2,3,4,5,6,7,8\}$ and partitions $P = \{\{1,3,5\},\{2,4,6\},\{7,8\}\}$ and $Q = \{\{1,3,5\},\{2,4,6\},\{7,8\}\}$. Then $P \Rightarrow_1 Q$ and $Q \Rightarrow_1 P$.
So, $Cons(P,Q) = 1$.
(iii) Let $U = \{1,2,3,4,5,6,7,8\}$ and partitions $P = \{\{1,4,5\},\{2,8\},\{6,7\},\{3\}\}$ and $Q = \{\{1,3,5\},\{2,4,7,8\},\{6\}\}$. Then $P \Rightarrow_{\frac{3}{8}} Q$ and $Q \Rightarrow_{\frac{1}{8}} P$.
So, $Cons(P,Q) = \frac{\frac{3}{8}+\frac{1}{8}+n\frac{3}{8}\frac{1}{8}}{n+2}$, where n is a non-negative integer.

Although any choice of n satisfies the initial requirements, some special values for it may be of special significance e.g $n = 0$, $n = Card(U)$ and n as defined in proposition 5. We shall make discussions on two of such values latter. 'n' shall be referred to as the 'consistency constant' or simply 'constant' in the sequel. The constant is a kind of constraint on consistency measure as shown in the next proposition.

Proposition 8. *For two knowledges P and Q if $n_1 \leq n_2$ then $Cons_1(P,Q) \geq Cons_2(P,Q)$ where $Cons_i(P,Q)$ is the consistency degree when n_i is the constant taken.*

Proof. Let $P \Rightarrow_a Q$ and $Q \Rightarrow_b P$. Since $n_1 \leq n_2$, so, $n_2 - n_1 \geq 0$. So $Cons_1(P,Q) = \frac{a+b+n_1ab}{n_1+2}$ and $Cons_2(P,Q) = \frac{a+b+n_2ab}{n_2+2}$. Now, $\frac{a+b+n_1ab}{n_1+2}$ - $\frac{a+b+n_2ab}{n_2+2}$ $= \frac{(n_2-n_1)(a+b-2ab)}{(n_1+2)(n_2+2)} \geq 0$ iff $(n_2 - n_1)(a + b - 2ab) \geq 0$ iff $(a + b - 2ab) \geq 0$ iff $a + b \geq 2ab$. Now, $\frac{a+b}{2} \geq \sqrt{ab} \geq ab$. So $a + b \geq 2ab$ holds. This shows that $Cons_1(P,Q) \geq Cons_2(P,Q)$. □

Proposition 9. *If n = the number of elements $a \in U$ such that $[a]_P \not\subseteq [a]_Q$ and $[a]_Q \not\subseteq [a]_P$, then $n = CardU$ - [Card $\bigcup_{X\in U/Q} \underline{P}X$ + Card $\bigcup_{X\in U/P} \underline{Q}X$ - Card($\bigcup_{X\in U/Q} \underline{P}X \bigcap \bigcup_{X\in U/P} \underline{Q}X$)].*

Proof. Here the number of elements $a \in U$ such that $[a]_P \subseteq [a]_Q$ = Card $\bigcup_{X\in U/Q}$ $\underline{P}X$ ...(i). Now the number of elements a$\in U$ such that $[a]_Q \subseteq [a]_P$=Card $\bigcup_{X\in U/P}$ $\underline{Q}X$ ...(ii). So the number of elements common to (i) and (ii) = Card($\bigcup_{X\in U/Q}$ $\underline{P}X$ $\bigcap \bigcup_{X\in U/P}$ $\underline{Q}X$)] ...(iii) . From (i), (ii) and (iii) the proposition follows. □

One can observe that the definition of a consistent object in [5,7] may be generalized relative to any pair (P,Q) of partitions of the Universe, not only restricted to the partitions caused due to the pair (CON, DEC) where CON is the set of condition attributes and DEC is the decision attributes. With this extension of the notion, n is the count of all those objects a such that a is not consistent relative to both the pairs (P,Q) and (Q,P). In the following examples n is taken to be this number.

Example 6. (i) Let $U = \{1,2,3,4,5,6,7,8\}$ and partitions $P = \{\{1,3,5\},\{2,4,6\},\{7,8\}\}$ and $Q = \{\{1,2,7\},\{3,4,8\},\{5,6\}\}$. Then $P \Rightarrow_0 Q$ and $Q \Rightarrow_0 P$. Here $n = 8$. So, $Cons(P,Q) = \frac{0+0+8.0.0}{8+2} = 0$.
(ii) Let $U = \{1,2,3,4,5,6,7,8\}$ and partitions $P = \{\{1,3,5\},\{2,4,6\},\{7,8\}\}$ and $Q = \{\{1,3,5\},\{2,4,6\},\{7,8\}\}$. Then $P \Rightarrow_1 Q$ and $Q \Rightarrow_1 P$. Here $n = 0$. So, $Cons(P,Q) = \frac{1+1+0.1.1}{0+2} = 1$.
(iii) Let $U = \{1,2,3,4,5,6,7,8\}$ and partitions $P = \{\{1,4,5\},\{2,8\},\{6,7\},\{3\}\}$ and $Q = \{\{1,3,5\},\{2,4,7,8\},\{6\}\}$. Then $P \Rightarrow_{\frac{3}{8}} Q$ and $Q \Rightarrow_{\frac{1}{8}} P$. Here $n = 4$. So, $Cons(P,Q) = \frac{\frac{3}{8}+\frac{1}{8}+4.\frac{3}{8}.\frac{1}{8}}{4+2} = \frac{11}{96}$.

If the t-norm is taken to be $max(0, a + b - 1)$, then the corresponding s-norm is $min(1, a + b)$. For the t-norm $min(a, b)$, the s-norm is $max(a, b)$. There is an order relation in the t-norms/ s-norms, viz.

any t-norm $\leq min \leq max \leq$ any s-norm.

In particular

$max(0, a+b-1) \leq min(a,b) \leq max(a,b) \leq min(1, a+b)$.

Where does the Cons function situate itself in this chain - might be an interesting and useful query. The following proposition answers this question.

Proposition 10. *$max(0, a+b-1) \leq Cons(P,Q) \leq max(a,b)$ if $P \Rightarrow_a Q$ and $Q \Rightarrow_b P$.*

To compare $Cons(P,Q)$ and $min(a,b)$, we have,

Proposition 11. *Let P and Q be two knowledges and $P \Rightarrow_a Q$ and $Q \Rightarrow_b P$. Then (i) $a = b = 1$ iff $min(a,b) = Cons(P,Q) = 1$,*
(ii) If either $a = 1$ or $b = 1$ then $min(a,b) \leq Cons(P,Q)$,
(iii) $min(a,b) = a \leq Cons(P,Q)$ iff $n \leq \frac{a-b}{a(b-1)}, a \neq 0, b \neq 1$,
(iv) $min(a,b) = a \geq Cons(P,Q)$ iff $n \geq \frac{a-b}{a(b-1)}, a \neq 0, b \neq 1$,
(v) $max(0, a+b-1) \leq Cons(P,Q) \leq max(a,b) \leq s(a,b) = min(1, a+b)$.

The Cons function seems to be quite similar to a t-norm but not the same. So a closer look into the function is worthwhile.

We define a function $\star : [0,1] \times [0,1] \to [0,1]$ as follows $\star(a,b) = \frac{a+b+nab}{n+2}$ where n is a non-negative integer.

Proposition 12. *(i) $0 \leq \star(a,b) \leq 1$,*
(ii) If $a \leq b$ then $\star(a,b) \leq \star(a,c)$,
(iii) $\star(a,b) = \star(b,a)$,
(iv) $\star(a, \star(b,c)) = \star(\star(a,b), c)$ iff $a = c$;
$\star(a, \star(b,c)) \leq \star(\star(a,b), c)$ iff $a \leq c$;
$\star(a, \star(b,c)) \geq \star(\star(a,b), c)$ iff $a \geq c$,
(v) $\star(a,1) \geq a$, equality occurring iff $a = 1$,
(vi) $\star(a,0) \leq a$, equality occurring iff $a = 0$,
(vii) $\star(a,b) = 1$ iff $a = b = 1$ and $\star(a,b) = 0$ iff $a = b = 0$,
(viii) $\star(a,a) = a$ iff either $a = 0$ or $a = 1$,

The consistency function *Cons* gives a measure of similarity between two knowledges. It would be natural to define a measure of inconsistency or dissimilarity now. In [6] a notion of distance is available.

Definition 14. *If $P \Rightarrow_a Q$ and $Q \Rightarrow_b P$ then the distance function is denoted by $\rho(P,Q)$ and defined as $\rho(P,Q) = \frac{2-(a+b)}{2}$.*

Proposition 13. *The distance function ρ satisfies the conditions :*
(i) $o \leq \rho(P,Q) \leq 1$
(ii) $\rho(P,P) = 0$
(iii) $\rho(P,Q) = \rho(Q,P)$
(iv) $\rho(P,R) \leq \rho(P,Q) + \rho(Q,R)$.

For proof the reader is referred to [6].

Definition 15. *We now define a measure of inconsistency by:*

$InCons(P,Q)$ = 1 - $Cons(P,Q)$

Proposition 14. *(i)* $o \leq InCons(P,Q) \leq 1$,
(ii) $InCons(P,P) = 0$,
(iii) $InCons(P,Q) = InCons(Q,P)$,
(iv) $InCons(P,R) \leq InCons(P,Q) + InCons(Q,R)$ *for a fixed constant* n.

Proof. of (iv) : Let $P \Rightarrow_x R$, $R \Rightarrow_y P$, $P \Rightarrow_a Q$, $Q \Rightarrow_b P$, $Q \Rightarrow_l R$, $R \Rightarrow_m Q$ ...(i). Now $InCons(P,R) = \frac{n+2-x-y-nxy}{n+2} \leq InCons(P,Q) + InCons(Q,R) = \frac{n+2-a-b-nab}{n+2} + \frac{n+2-l-m-nlm}{n+2} = \frac{2(n+2)-n(ab+lm)-(a+b+l+m)}{n+2}$
iff $n+2-x-y-nxy \leq 2(n+2) - n(ab+lm) - (a+b+l+m)$
iff $n(ab+lm-xy-1) \leq 2+x+y-(a+b+l+m)$...(ii).

From (i) by Proposition 3(v) we have $x \geq (a+m-1)$ and $y \geq (b+l-1)$.

Hence $(ab+lm-xy-1) \leq (ab+lm-(a+m-1)(b+l-1)-1) = (a(1-l)+b(1-m)+(m-1)+(l-1)) \leq (1-l+1-m+m-1+l-1)$ (because $0 \leq a,b \leq 1) = 0$. ...(iii) Now, $2+x+y-(a+b+l+m) = 2(\frac{2-a-b}{2}+\frac{2-l-m}{2}-\frac{2-x-y}{2})$ $= 2(\rho(P,Q)+\rho(Q,R)-\rho(P,R)) \geq 0$. ...(iv)[by Proposition 13(iv)].

Thus the left hand side of inequality (ii) is negative and the right hand side of (ii) is positive. So (iv) i.e triangle inequality is established. □

Proposition 11 shows that for any fixed n the inconsistency measure of knowledge is a metric. It is also a generalization of the distance function ρ in [6]; InCons reduces to ρ when $n = 0$. n is again a kind of constraint on the inconsistency measure - as n increases, the inconsistency increases too.

4.1 Consistency Degree w.r.t Covering

Definition 16. *We define consistency degree in the same way :* $Cons^i(\mathcal{C}_1,\mathcal{C}_2) = \frac{a+b+nab}{n+2}$ *where* $Dep^i(C_1,C_2) = a$ *i.e.,* $\mathcal{C}_1 \Rightarrow_a \mathcal{C}_2$ *and* $Dep^i(C_2,C_1) = b$ *i.e.,* $\mathcal{C}_2 \Rightarrow_b \mathcal{C}_1$ *where* $i = 1,2,3,4$.

Example 7. Let $\mathcal{C}_1 = \{\{1,2,3\},\{3,4,8\},\{6,7,8\},\{8,9\}\}$ and $\mathcal{C}_2 = \{\{1,2,3,4\},\{5,8\},\{6,7\},\{8,9\}\}$.

Then $Dep^1(\mathcal{C}_1,\mathcal{C}_2) = \frac{1}{3}$, $Dep^2(\mathcal{C}_1,\mathcal{C}_2) = \frac{1}{3}$, $Dep^3(\mathcal{C}_1,\mathcal{C}_2) = \frac{5}{9}$, $Dep^4(\mathcal{C}_1,\mathcal{C}_2) = 1$.
Also, $Dep^1(\mathcal{C}_2,\mathcal{C}_1) = \frac{1}{3}$, $Dep^2(\mathcal{C}_2,\mathcal{C}_1) = \frac{1}{3}$, $Dep^3(\mathcal{C}_2,\mathcal{C}_1) = \frac{4}{9}$, $Dep^4(\mathcal{C}_2,\mathcal{C}_1) = \frac{5}{9}$.
So, $Cons^i(\mathcal{C}_1,\mathcal{C}_2)$ for $i = 1,2,3,4$ are as follows :

$Cons^1(\mathcal{C}_1,\mathcal{C}_2) = \frac{\frac{1}{3}+\frac{1}{3}+n.\frac{1}{3}.\frac{1}{3}}{n+2} = \frac{n+6}{9(n+2)}$,
$Cons^2(\mathcal{C}_1,\mathcal{C}_2) = \frac{\frac{1}{3}+\frac{1}{3}+n.\frac{1}{3}.\frac{1}{3}}{n+2} = \frac{n+6}{9(n+2)}$,
$Cons^1(\mathcal{C}_1,\mathcal{C}_2) = \frac{\frac{5}{9}+\frac{4}{9}+n.\frac{5}{9}.\frac{4}{9}}{n+2} = \frac{20n+81}{81(n+2)}$,
$Cons^1(\mathcal{C}_1,\mathcal{C}_2) = \frac{1+\frac{5}{9}+n.1.\frac{5}{9}}{n+2} = \frac{5n+14}{9(n+2)}$.

Observation

(a) $Cons^i(\mathcal{C}_1,\mathcal{C}_2) = 1$ iff $Dep^i(\mathcal{C}_1,\mathcal{C}_2) = 1$ and $Dep^i(\mathcal{C}_2,\mathcal{C}_1) = 1$.

Its interpretations for $i = 1, 2, 3, 4$ are given by:

$Cons^1(\mathcal{C}_1, \mathcal{C}_2) = 1$ iff $\forall x \in U, N_x^{\mathcal{C}_1} \subseteq X$ for some $X \in \mathcal{C}_2$ and $\forall x \in U, N_x^{\mathcal{C}_2} \subseteq X$ for some $X \in \mathcal{C}_1$.

$Cons^2(\mathcal{C}_1, \mathcal{C}_2) = 1$ iff $\forall x \in U, N_x^{\mathcal{C}_1} \subseteq X$ for some $X \in \mathcal{C}_2$ and $\forall x \in U, N_x^{\mathcal{C}_2} \subseteq X$ for some $X \in \mathcal{C}_1$.

$Cons^3(\mathcal{C}_1, \mathcal{C}_2) = 1$ iff each $C_i (\in \mathcal{C}_1) \subseteq X$, for some $X \in \mathcal{C}_2$ and each $C_i (\in \mathcal{C}_2) \subseteq X$, for some $X \in \mathcal{C}_1$

$Cons^4(\mathcal{C}_1, \mathcal{C}_2) = 1$ iff for all x, $P_x^{\mathcal{C}_1} \subseteq X$ for some $X \in \mathcal{C}_2$ and for all x, $P_x^{\mathcal{C}_2} \subseteq X$ for some $X \in \mathcal{C}_1$

(b) $Cons^i(\mathcal{C}_1, \mathcal{C}_2) = 0$ iff $Dep^i(\mathcal{C}_1, \mathcal{C}_2) = 0$ and $Dep^i(\mathcal{C}_2, \mathcal{C}_1) = 0$.

So, the interpretations are:

$Cons^1(\mathcal{C}_1, \mathcal{C}_2) = 0$ iff $\forall x \in U$, there does not exists any $X \in \mathcal{C}_2$ such that $N_x^{\mathcal{C}_1} \subseteq X$ and $\forall x \in U$, there does not exists any $X \in \mathcal{C}_1$ such that $N_x^{\mathcal{C}_2} \subseteq X$.

$Cons^2(\mathcal{C}_1, \mathcal{C}_2) = 0$ iff $\forall x \in U$, there does not exists any $X \in \mathcal{C}_2$ such that $N_x^{\mathcal{C}_1} \subseteq X$ and $\forall x \in U$, there does not exists any $X \in \mathcal{C}_1$ such that $N_x^{\mathcal{C}_2} \subseteq X$.

$Cons^3(\mathcal{C}_1, \mathcal{C}_2) = 0$ iff for any $C_i \in \mathcal{C}_1$ there does not exists any $X \in \mathcal{C}_2$ such that $C_i \subseteq X$ and for any $C_i \in \mathcal{C}_2$ there does not exists any $X \in \mathcal{C}_1$ such that $C_i \subseteq X$

$Cons^4(\mathcal{C}_1, \mathcal{C}_2) = 0$ iff for all x, there does not exists any $X \in \mathcal{C}_2$ such that $P_x^{\mathcal{C}_1} \subseteq X$ and for all x, there does not exists any $X \in \mathcal{C}_1$ such that $P_x^{\mathcal{C}_2} \subseteq X$

Definition 17. *A measure of inconsistency for the case of covering in the same way is defined as follows :*

$InCons(P, Q)$ = 1 - $Cons(P, Q)$.

5 Towards a Logic of Consistency of Knowledge

We are now at the threshold of a logic of consistency (of knowledge). Along with the usual propositional connectives the language shall contain two binary predicates, '*Cons*' and '*Dep*' for consistency and dependency respectively. At least the following features of this logic are present.

(i) $0 \leq Cons(P, Q) \leq 1$,
(ii) $Cons(P, P) = 1$,
(iii) $Cons(P, Q) = Cons(Q, P)$,
(iv) $Cons(P, Q) = 0$ iff $Dep(P, Q) = 0$ and $Dep(Q, P) = 0$

and $Cons(P, Q) = 1$ iff $Dep(P, Q) = 1$ and $Dep(Q, P) = 1$
In case P,Q,R partitions we also get
(v) $Cons(P, Q)$ and $Cons(Q, R)$ implies $Cons(P, R)$.

(i) shows that the logic is many-valued; (ii) and (iii) are natural expectations; (iv) conforms to assumptions 1 and 2 (section2); (v) shows transitivity the predicate Cons in the special case of partitions.

That the transitivity holds is shown below. We want to show that $Cons(P, Q)$ and $Cons(Q, R)$ implies $Cons(P, R)$ i.e, $Cons(P, Q)$ and $Cons(Q, R) \leq Cons(P, R)$. We use Lukasiewicz t-norm to compute 'and'. Let n be the fixed constant. So,what is needed is $Max(0, Cons(P, Q) + Cons(Q, R) - 1) \leq Cons(P, R)$.

Clearly, $Cons(P, R) \geq 0$ ...(i).

We shall now show $Cons(P, R) \geq Cons(P, Q) + Cons(Q, R) - 1$. Let $P \Rightarrow_x R$, $R \Rightarrow_y P$, $P \Rightarrow_a Q$, $Q \Rightarrow_b P$, $Q \Rightarrow_l R$, $R \Rightarrow_m Q$ So $x \geq (a + m - 1)$ and $y \geq (b + l - 1)$ [cf. Proposition 3(v)]...(ii).

So, $Cons(P, Q) + Cons(Q, R) - 1 = \frac{a+b+nab}{n+2} + \frac{l+m+nlm}{n+2} - 1$
$= \frac{(a+l-1)+(b+m-1)+n(ab+lm-1)}{n+2} \leq \frac{x+y+n(ab+lm-1)}{n+2}$ [using (ii)]...(iii).

Here, $xy \geq (a+l-1)(b+m-1) = ab+lm+(m-1)(a-1)+(b-1)(l-1)-1 \geq ab+lm-1$. [as, $m-1 \leq 0$, $a-1 \leq 0$, so $(m-1)(a-1) \geq 0$, and $b-1 \leq 0$, $l-1 \leq 0$, $(b-1)(l-1) \geq 0$] ...(iv) . So (iii) and (iv) imply $Cons(P, Q)+Cons(Q, R)-1 \leq \frac{x+y+nxy}{n+2} = Cons(P, R)$ ... (v).

(i)-(v) pave the way of formulating axioms of a possible logic of knowledge.

6 Concluding Remarks

This paper is only the beginning of a research on a many valued logic of dependency and consistency of knowledges where knowledge is in the context of incomplete information understood basically as proposed by Pawlak. Various ways of defining lower and upper approximations indicate that the modalities are also different and hence corresponding logics would also be different. We foresee interesting logics being developed and significant applications of the concepts *Dep*, *Cons* and the the operator $\star$.

Acknowledgement

The first author acknowledges the financial support from the University Grants Commission, Government of India.

References

1. Bianucci, D., Cattaneo, G., Ciucci, D.: Entropies and co-entropies of coverings with application to incomplete information systems. Fundamenta Informaticae 75, 77–105 (2007)
2. Cattanio, G., Cucci, D.: Lattice Properties of Preclusive and Classical Rough Sets. Personal Collection

3. Chakraborty, M.K., Samanta, P.: Consistency-Degree Between Knowledges. In: Kryszkiewicz, M., et al. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 133–141. Springer, Heidelberg (2007)
4. Klir, G.J., Yuan, B.: Fuzzy Sets And Fuzzy Logic: Theory and Applications. Prentice-Hall of India, Englewood Cliffs (1997)
5. Nguyen, N.T., Malowiecki, M.: Consistency Measures for Conflict Profiles. In: Peters, J.F., Skowron, A., Grzymała-Busse, J.W., Kostek, B.z., Świniarski, R.W., Szczuka, M.S. (eds.) Transactions on Rough Sets I. LNCS, vol. 3100, pp. 169–186. Springer, Heidelberg (2004)
6. Novotný, M., Pawlak, Z.: Partial Dependency of Attributes. Bull. Polish Acad. of Sci., Math. 36, 453–458 (1988)
7. Pawlak, Z.: Rough Sets. Internal Journal of Information and Computer Science 11, 341–356 (1982)
8. Pawlak, Z.: On Rough Dependency of Attributes in Information System. Bull. Polish Acad. of Sci., Math. 33, 551–559 (1985)
9. Pawlak, Z.: ROUGH SETS - Theoritical Aspects of Reasoning About Data. Kluwer Academic Publishers, Dordrecht (1991)
10. Pomykala, J.A.: Approximation, Similarity and Rough Constructions. ILLC Prepublication Series for Computation and Complexity Theory CT-93-07. University of Amsterdam
11. Qin, K., Gao, Y., Pei, Z.: On Covering Rough Sets. In: Yao, J.T., et al. (eds.) RSKT 2007. LNCS, vol. 4481, pp. 34–41. Springer, Heidelberg (2007)
12. Sakai, H., Okuma, A.: Basic Algorithm and Tools for Rough Non-deterministic Information Analysis. In: Peters, J.F., Skowron, A., Grzymała-Busse, J.W., Kostek, B.z., Świniarski, R.W., Szczuka, M.S. (eds.) Transactions on Rough Sets I. LNCS, vol. 3100, pp. 209–231. Springer, Heidelberg (2004)
13. Skowran, A., Stepaniuk, J.: Tolerance approximation spaces. Fundamenta Informaticae 27, 245–253 (1996)
14. Slezak, D., Wasilewski, P.: Granular Sets - Foundations and Case Study of Tolerance Spaces. In: An, A., et al. (eds.) RSFDGrC 2007. LNCS, vol. 4482, pp. 435–442. Springer, Heidelberg (2007)
15. Yao, Y.: Semantics of Fuzzy Sets in Rough Set Theory. In: Peters, J.F., et al. (eds.) Transactions on Rough Sets II. LNCS, vol. 3135, pp. 297–318. Springer, Heidelberg (2004)
16. Zakowski, W.: Approximation the space (u, π). Demonstratio Mathematica 16, 761–769 (1983)

A New Approach to Distributed Algorithms for Reduct Calculation*

Tomasz Strąkowski and Henryk Rybiński

Warsaw Uniwersity of Technology, Poland
T.Strakowski@elka.pw.edu.pl,
H.Rybinski@ii.pw.edu.pl

Abstract. Calculating reducts is a very important process. Unfortunately, the process of computing all reducts in NP-hard. There are a lot of heuristic solutions for computing reducts, but they do not guarantee achieving complete set of reducts. We propose here three versions of an exact algorithm, designed for parallel processing. We present here how to decompose the problem of calculating reducts, so that parallel calculations are efficient.

Keywords: Rough set theory, reducts calculations, distributed computing.

1 Introduction

Nowadays, the ability of collecting data is much higher than the ability of processing them. Rough Set Theory (RST) provides means for discovering knowledge from data. One of the main concepts in RST is the notion of reduct, which can be seen as a minimal set of conditional attributes preserving the required classification features [1]. In other words, having a reduct of a decision table we are able to classify objects (i.e. take decisions) with the same quality as with all attributes. However, the main restriction in practical use of RST is that computing all reducts is NP-hard. It is therefore of high importance to find out efficient algorithms that compute reducts efficiently.

There are many ideas how to speedup computing of reducts [2], [3], [4], [5]. Many of the presented algorithms are based on some heuristics. The disadvantage of the heuristic solution is that it does not necessary give us a complete set of reducts, in addition, some results can be over-reducts. Another way to speed up the calculation processes, not yet explored sufficiently, could be distributing the computations over a set of processors, and perform the calculations in parallel.

* The research has been partially supported by grant No 3 T11C 002 29, received from Polish Ministry of Education and Science, and partially supported by grant received from rector of Warsaw University of Technology No 503/G/1032/4200/000.

In this paper we analyze how to speed up the calculations of the complete sets of reducts by distributing the processing over a number of available processors.

A parallel version of a genetic algorithm for computing reducts has been presented in [3]. The main disadvantage of this approach is that the algorithm does not necessary find all the reducts. In this paper we present various types of the problem decomposition for calculating reducts. We present here three versions of distributing the processing, each of them generating all the reducts of a given information system. We will also discuss the conditions for decomposing the problem, and present criteria that enable one to find out the best decomposition.

The paper is composed as follows. In Section 2 we recall basic notions related to the rough set theory, and present the analogies between finding reducts in RST and the transformations of logical clauses. We also present a naïve algorithm for finding a complete set of reducts and discuss the complexity of the algorithm. In Section 3 we present 3 various ways of decomposing the process of reduct calculations. Section 4 is devoted to experimental results, performed with all three proposed approaches. We conclude the paper with a discussion about the effectiveness of the approaches and their areas of applications.

2 Computing Reducts and Logic Operations

Let us start with recalling basic notions of the rough set theory. In practical terms, knowledge is coded in an information system (*IS*). *IS* is a pair (*U*,*A*) where U is finite set of elements, and A is a finite set of attributes which describe each element. For every $a \in A$ there is a function $U \rightarrow V_a$, assigning a value $v \in V_a$ of the attribute a to the objects $u \in U$, where V_a is domain of a. The indiscernibility relation is defined as follows:

$$IND(A) = \{(u, v) : u, v \in U, a(u) = a(v), a \in A\}$$

Informally speaking, two objects u and v are indiscernible for the attribute a if they have the same value of that attribute. The indiscernibility relation could be defined for the set of attributes $IND(B) = \bigcap_{a \in B} IND(a), B \subseteq A$. One of the most important ideas in RST is the notion of reduct.

Reduct is a minimal set of attributes B, $B \subseteq A$, for which the indiscernibility relation in U is exactly the same, as for the set A, i.e. $IND(B) = IND(A)$. Superreduct is a super set of a reduct.

Given a set of attributes B, $B \subseteq A$, we define a *B-related* reduct as a set C of attributes, $B \cap C = \emptyset$, which preserves the partition of $IND(B)$ over U.

Given $u \in U$, we define *local reduct* as a minimal set of attributes capable of distinguishing this particular object from the other objects, as well, as the total set of attributes. Let us introduce a discernibility function (denoted by $disc(B, u)$) as a set of all object v discernible with u for the set of attributes B:

$$disc(B, u) = \{v \in U | \forall a \in B(a(u) \neq a(v))\}$$

Table 1. Decision Table

	a	b	c	d
u_1	1	2	3	1
u_2	1	2	1	2
u_3	2	2	3	2
u_4	2	2	3	2
u_5	3	5	1	3

Table 2. Indiscernibility matrix

	u_1	u_2	u_3	u_4	u_5
u_1		c	a	a	abc
u_2	c		ac	ac	ab
u_3	a	ac			abc
u_4	a	ac			abc
u_5	abc	ab	abc	abc	

Table 3. Interpretation of the indiscernibility matrix

	Discernibility Function	*CNF form (after reduction)*	*DNF Formula (Prime Implicants)*	*Local reducts*
u_1	$c \wedge a \wedge (a \vee b \vee c)$	$c \wedge a$	$a \vee a$	{a,c}
u_2	$c \wedge (a \vee c) \wedge (a \vee b)$	$c \wedge (a \vee b)$	$(a \wedge c) \vee (b \wedge c)$	{a,c};{b,c}
u_3	$a \wedge (a \vee c) \wedge (a \vee b)$	a	a	{a}
u_4	$a \wedge (a \vee c) \wedge (a \vee b)$	a	a	{a}
u_5	$(a \wedge b \wedge c) \vee (a \wedge b)$	$(a \wedge b)$	$a \wedge b$	{a };{b }

Local reduct for the element $u \in U$ is a minimal set of attributes B, $B \subseteq A$, such that *disc(B,u)* = *disc(A,u)*. Now, let us show some similarities between reducts and some logic operation. The relationships between reducts and logical expressions were first presented in [6]. Let us consider a decision table, as in Table 1. We have here five elements ($u_1 - u_5$), three conditional attributes, namely a, b, c, and one decision attribute d. The indiscernibility matrix for this table is shown in Table 2: The interpretation of the above indiscernibility matrix can be presented in the form of Table 3: The *i-th* row shows here the following: in column *1* there is a rule saying which attributes have to be used to discern the *i-th* object (u_i) with any other objects of IS from Table 1 (discernibility). The second column shows the same rule in the form of CNF (after reduction), the 3^{rd} one presents the rule in disjunctive normal form (*DNF*), whereas the last column provides the local reducts for u_i.

Algorithm 2.1. Reduct Set Computation(DT)

```
$Compute\ Indiscernibility\ Matrix\ M(A) = (C_{ij})$
$Transform\ M\ to\ one\ dimensional\ table\ T$
$Reduce\ T\ using\ absorption\ laws$
comment: from CNF to prime implicant
$Sort\ T$
comment: Sorting is our modification, d - number of elements in T
$build\ the\ families\ of\ R_1,\ R_2,\ \ ,\ Rn\ in\ the\ following\ way:$
    $R_0 = \emptyset$
    for $i \leftarrow 0$ to $d$
        do
            if $Stop\ condition\ is\ true$ comment: It is our modification
              then $Break\ algorithm R_d = R_i$
              else $R_i = S_i \cup K_i\ where\ S_i = \{r \in R_{i-1} : r \cap T_i \neq \emptyset\}$
            $K_i = (r \cup \{a\}) \forall a \in T_i, r \in R_{i-1} : r \cap T_i = \emptyset$
$Remove\ redundant\ elements\ from\ R_d$
$Remove\ Super - Reducts$
$RED(a) = R_d$
return $(RED(A))$
```

Now let us recall a naïve method for calculating reducts. It is a slight modification of the algorithm presented in [2], and is given abowe in the form of a pseudo code. This code differs from the original one in two places. Fist, we sort the clauses in discernibility function by length (the shortest clauses are first). Then, we change here the stop condition: T_i is the *i-th* clause of *prime implicant*. R_i is the set of candidate reducts. In the *i-th* step we check $r \in R_i$, and if $r \cap Ti \neq \emptyset$, then $R_{i+1} := R_{i+1} \cup r$, otherwise the clause is split onto separate attributes, and each attribute is added to r, making a new reduct candidate to be included to R_{i+1}. As the clauses T_i are sorted, we can stop the algorithm when $k + l_i > |A|$, where k is the length of the shortest candidate reduct, l_i is the length of T_i.

Let us reconsider the time and space complexities of the naïve algorithm. There are four sequential parts in the algorithm: (1) generating the indiscernibility matrix (*IND* matrix); (2) converting the matrix to discernibility function (using absorption laws); and (3) converting to the *DNF* form (prime implicants), i.e. reducts.

The *IND* matrix is square and symmetric (with null values on the diagonal). The size of the matrix is $|U| \times |U|$, $|U|$ denotes the number of the elements in DT. So, the time and space complexities are:

$$O(\frac{|U|^2 - |U|}{2}) \tag{1}$$

The complexity of the process of converting from *IND* to the discernibility function formulae is linear, so can be ignored. The complexity of the conversion from discernibility function to *CNF* is $O(n^2)$ (in the worst case), where n is the number of clauses in the discernibility function. No additional data structures are needed, so the space complexity can be ignored.

The hardest to estimate is the complexity of converting from *CNF* to *DNF*. The space complexity is estimated as:

$$O\binom{|A|}{\frac{|A|}{2}} \tag{2}$$

It is the maximal number of the candidate reducts in the conversion process. The proof on the maximal numbers of reducts was presented in [4]. More complicated is to estimate the time complexity. Given n as the number of clauses in the *discernibility function*, we can estimate it as:

$$O(\binom{|A|}{\frac{|A|}{2}} \times n) \tag{3}$$

During the conversion process from discernibility function to prime implicants in one step we compare every candidate *reduct* with the *i-th* clause of discernibility function. The number of steps is equal to the number of clauses. The maximum number of clauses in the *discernibility function* is: $n = \frac{|U|^2 - |U|}{2}$ (in the worst case, where the absorption laws cannot be used). Hence, the time complexity is:

$$O(\binom{|A|}{\frac{|A|}{2}} \times \frac{|U|^2 - |U|}{2}) \tag{4}$$

Let us summarize now our considerations:

1. The maximal space requirement depends only on the number of the attributes in DT.
2. The time of computing *IND* depends polynomialy on the number of objects in DT.
3. The time of computing all the reducts depends exponentially on the number of attributes (for a constant number of objects).

The exponential explosion of the complexity appears in the last part of the algorithm, during the conversion from *CNF* to prime implicants. The best practice is to decompose the more complex part, though it is not the only possible place. Sometimes computing *IND* is more time consuming than evaluating the conversions. We will discuss the options in the next section.

3 Decomposition of the Problem

There are several ways of decomposing the algorithm. One possibility is to split DT, compute reducts for each part independently and merge the results. Another idea is to compute *IND* matrix sequentially, convert it to *discernibility function* and *CNF*, and then split *discernibility function* into several parts to be calculated separately, so that the conversions to *DNF* are made in the separate nodes of the algorithm, and then the final result is obtained by merging the partial results.

The two proposals above consist in a horizontal decomposition, in this sense that we split the table *DT* into some sub-tables, and then use the partial results to compute the final reduct. Certainly, the partial results are not necessarily reducts. They can be though related reducts, and additional (post)-processing is needed to calculate reducts. Both proposals will be described in more detail in this Section, in 3.1 and 3.2 respectively.

In the paper we propose yet another solution, based on a vertical decomposition. In particular, during the process of converting *CNF* to *DNF* we split the set of candidate reducts among a number of processors, which then serve in parallel for processing the consecutive *clauses*. We call this decomposition vertical because it splits the set of candidate reducts (subsets of the attributes) into separate subsets, instead of splitting the set of objects. For each subset of the candidate reducts, the conversion is completed in a separate node of algorithm (processor).

Let us note that every candidate reduct passes comparisons with every clause. This will give us a guarantee that the partial results in each node are reducts or super reducts. Having computed the partial results, in the last phase of the algorithm we join them to the final reducts set.

Let us also note that there is a difference between using partial results obtained from horizontal and vertical decompositions. In the first case we have to merge partial reducts, which is a complex, and time consuming process, whereas in the second case we have to join the partial results, and remove duplicates, and super reducts. This process is fairly simple. The third proposal is presented in p. 3.4. Below we describe the three proposals in more detail.

3.1 Splitting Decision Table

Let us present the process of decomposing DT. We split DT into two separate, randomly selected subsets: X_1 and X_2, and for each of them we compute the reducts. If now we would like to "merge" the results, the final result does not take into account indiscernibilities between objects from X_1 and X_2. It is therefore necessary to compute another part of the IND matrix to calculate the discernibility for the pairs $(x_i x_j)$, $x_i \in X_1$, and $x_j \in X_2$. In Fig. 1 it is shown how the decomposition of DT influences splitting the IND matrix (denoted by M). $M(X_k)$, $k = 1, 2$, are the parts related to discernibility of the object both from X_k. $M(X_1 \cup X_2)$ is a part of M with information about discernibility between x_i, x_j, such that $x_i \in X_1$, and $x_j \in X_2$. In this sense the decomposition of DT is not disjoint. However, in the sense of splitting M into disjoint parts, the decomposition is disjoint. We can thus conclude that for splitting DT into two sets we need three processing nodes. Similarly, if we split DT into three sets, we need six processing nodes. In general, if we split DT into n subsets we need $\frac{n^2+n}{2}$ processing nodes.

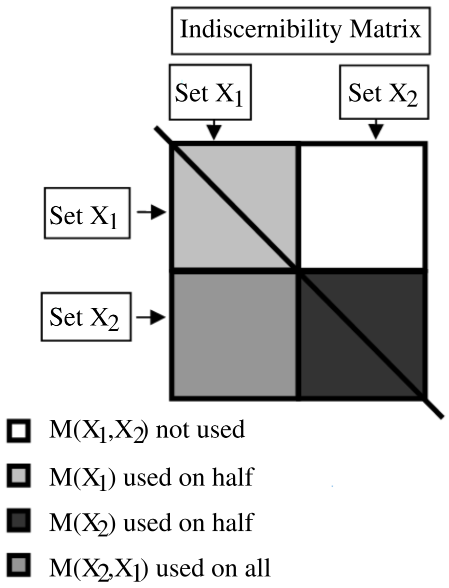


Fig. 1. Spliting DT

3.2 Spliting Discernibility Function

Another idea for decomposing the problem of computing all reducts is to split discernibility function into separate sections, and then to treat each section as a separate discernibility function. The conversion to *DNF* is made for every discernibility function, and then the partial results are merged as a multiplication of clauses. Let us illustrate it by the following example.

Example 1. Provided after applying the absorption laws we receive the discernibility function as below:

$$(a \vee b) \wedge (a \vee c) \wedge (b \vee d) \wedge (d \vee e)(*)$$

we can now convert it to the *DNF* form in the following sequential steps:

1. $(a \vee ac \vee ab \vee bc) \wedge (b \vee d) \wedge (d \vee e) = (a \vee bc) \wedge (b \vee d) \wedge (d \vee e)$
2. $(ab \vee ad \vee bcd \vee bc) \wedge (d \vee e) = (ab \vee ad \vee bc) \wedge (d \vee e)$
3. $(abd \vee abe \vee ad \vee ade \vee bcd \vee bce) = (ad \vee abe \vee bcd \vee bce)$

Instead of processing (*) sequentially let us split it into 2 parts:

1. $(a \vee b) \wedge (a \vee c)$
2. $(b \vee d) \wedge (d \vee e)$

The tasks (1) and (2) can be continued in 2 separate processing nodes, which leads to the forms:

1. $(a \vee ac \vee ab \vee bc) = (a \vee bc)$
2. $(bd \vee be \vee d \vee de) = (be \vee d)$

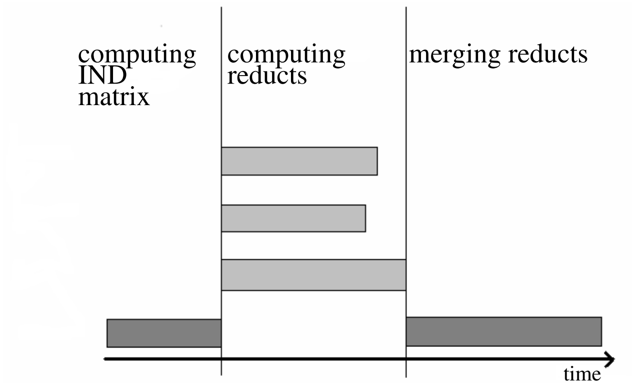


Fig. 2. The parallel processing with 3 nodes

Having the partial results from the nodes (1) and (2) we merge them: $(a \vee bc) \wedge (be \vee d)$ So we receive the final result: $(abe \wedge ad \wedge bce \wedge bcd)$ On Fig. 2 we present a general idea of processing algorithm in the parallel way, as sketched above. As one can see, in this approach we can split the calculations among as many nodes as many pairs of clauses we have in the *discernibility function* (obviously we can split the task to a smaller number of nodes, as well). There is though a final part of the algorithm, which is devoted to merging the partial results coming from the nodes.

This process is performed sequentially and its efficiency depends on the number of processing nodes. Obviously, we should avoid the cases when the cost of merging is higher than the savings from parallel processing. We discuss the issue in the next paragraph.

Merging of partial results

The process of merging the partial results is time consuming. It is equivalent to the process of finding Cartesian product of n sets, so the time requirement for this process depends on the number of the partial results, i.e. $O(\Pi|mi|)i = 1, 2, 3, ..n$, where $|mi|$ is the number of elements in the i^{th} partial result. There is though a way to perform also this process in a parallel way. Let us consider the case we have two partial results to merge - p_1 and p_2. We split p_1 into few separate subsets, so $p_1 = \bigvee_i p_{1i}$. Thus $p_1 \wedge p_2 = \bigvee_i p_{1i} \wedge p_2$, and each component $p_{1i} \wedge p_2$ can be processed in a separate processing node. The process of summing the partial conjunction results consists in removing duplicates and super reducts from the final result set. The more components of p_{1i} we have in p_1, the more processors we can use.

Optimal use of the processors

On Fig. 3 we present an example of using 5 processors for computing *reducts* by splitting *prime implicant.* We distinguish here four phases. The first one is for computing the IND matrix and *prime implicant* (marked by very light grey), then the conversion from *prime implicant* to *DNF* starts (light grey) on five nodes.

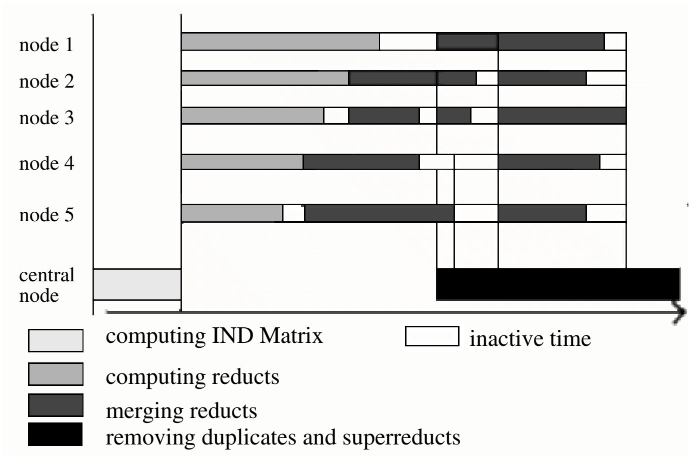


Fig. 3. Sample usage of processors for 5 nodes

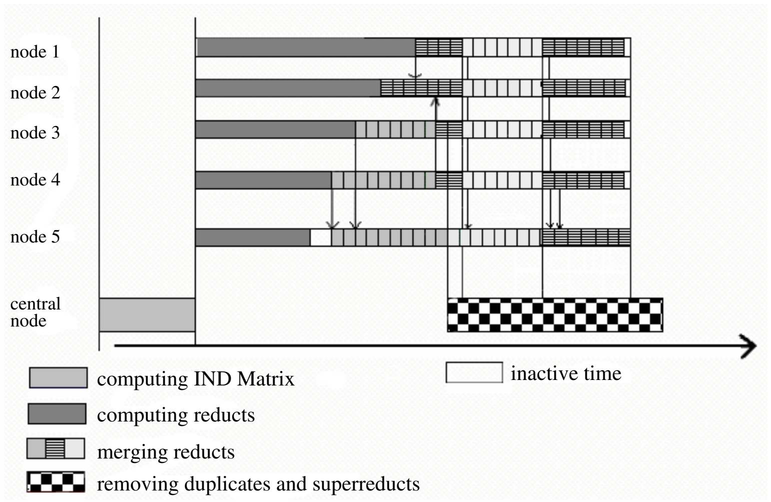


Fig. 4. Merging by bundles

When we have 2 conversions completed, the merging can start on the free nodes (dark grey). When any *partial reduct* results are provided, the final process of removing duplicates is performed sequentially (black). This solution is not optimal for the use of processors. There are a lot of periods where some nodes of the algorithm have to wait, even if some nodes have the same speed. The problem gets worse if some nodes differ in speed.

To solve this problem we propose in every merging of partial results P_1 and P_2 to split P_1 into more parts then we have free available processors. Thus, we decompose merging into many independent bundles. Each bundle can be processed asynchronously. Each processor processes as many bundles as it can.

In this case, the maximal time of waiting in every partial merging is the time of processing one bundle in the slowest node. Let us consider this proposal in more detail Fig. 4.

In this case the node N_3 does not have to wait for N_2, but it helps nodes N_4 and N_5 by merging bundles from P_4 and P_5. This task can be finished faster than in the previous example. After computing *DNF* from $\wedge(P_2, N_2)$ takes P_2 and P_3 from the queue, and starts computing set $\wedge(P_2, P_3)$. After computing $\wedge(P_4, P_5)$, the nodes N_3, N_4, N_5 join to N_2. Having finished P_1, the node N_1 takes the next task from the queue ($\wedge(P_1, P_4, P_5)$). Having finished processing $\wedge(P_2, P_3)$ the remaining free nodes join to the computations $\wedge(P_1, P_4, P_5)$. The last task is to compute $\wedge(P_1, P_2, P_3, P_4, P_5)$ by all the nodes.

3.3 Splitting Set of Candidate Reducts - Vertical Decomposition

Now we present the third way of decomposing calculations of *reducts*, which is the vertical one. The main idea is that during the conversion of *CNF* to *DNF* we split the formula into 2 parts across a (disjunctive) component. The idea of this decomposition was originally presented in [7]. Here we make a slight modification of this method. Let us go back again to the conversion process from *CNF* to *DNF*. Sequentially, the process can be performed as below:

1. $(\boldsymbol{a} \vee \boldsymbol{b}) \wedge (a \vee c) \wedge (b \vee d) \wedge (d \vee e)$
2. $(a \vee ac \vee ab \vee bc) \wedge (b \vee d) \wedge (d \vee e) = (\boldsymbol{a} \vee \boldsymbol{bc}) \wedge (b \vee d) \wedge (d \vee e)$
3. $(ab \vee ad \vee bcd \vee bc) \wedge (d \vee e) = (\boldsymbol{ab} \vee \boldsymbol{ad} \vee \boldsymbol{bc}) \wedge (d \vee e)$
4. $(abd \vee abe \vee ad \vee ade \vee bcd \vee bce) = (ad \vee abe \vee bcd \vee bce)$

The bold clauses $\boldsymbol{a}$ and $\boldsymbol{bc}$ relate to *"candidate reducts"*. Let us make the decomposition after the second step[1], and perform the process in two nodes:

Table 4. Decomposition of computation after second step

Node 1	*Node 2*
$(\boldsymbol{a}) \wedge (b \vee d) \wedge (d \vee e)$	$(\boldsymbol{bc}) \wedge (b \vee d) \wedge (d \vee e)$
$(ab \vee ad) \wedge (d \vee e)$	$(bc \vee bcd) \wedge (d \vee e) = (bc) \wedge (d \vee e)$
$(abd \vee abe \vee ad \vee ade) = (ad \vee abe)$	$(bcd \vee bce)$
$(ad \vee abe \vee bcd \vee bce)$	

The advantage of this decomposition is easiness of joining partial results - one should only add sets of reducts and remove super-reducts. This method reduces time of processing and space needed for storing *candidate reducts*. If we have one processor without enough memory for the *candidate reducts* we can decompose the process into two parts. The first part can be continued, whereas the second one can wait frozen, and restart after having finished the first one. This is more effective than using virtual memory, because the algorithm can

[1] It could have been done also after the 1st step, as well as after the 3^{rd} step.

decide what should be frozen, and what is executed. The disadvantage is that decomposition is done in the late phase of algorithm. It causes that the time saved by the decomposition can be inessential. Another disadvantage is that the algorithm depends on too many parameters. In particular, one has to choose a right moment to split the formula. In our experiment we have used the following rules:

1. do not split before doing 10% of the conversion steps;
2. the last split must be done before 60% of the conversion;
3. make splitting if the number of candidates is greater then u (u is a parameter).

The main difference between our proposal and the one presented in [7] is in spliting candidate of sets. In [7] it is proposed to split the set of candidates into n procesors after having the number of "candidates reducts" higher than branching factor [7]. The disadvantage of this approach is that we do not know the number of candidate reducts before completing computations, so it is hard to estimate the optimal value of the branching factor.

4 Experiments and Results

There are a some measures in the literature for the distributed algorithms. In our experiments we used two indicators:

1. Speedup
2. Efficiency

Following [8] we define speedup as $Sp = \frac{T1}{Tp}$, and efficiency as $Ep = \frac{Sp}{p}$, where T1 is the time of execution of the algorithm on one processor, Tp is the time needed by p processors, p is the number of processors.

We have tested all the presented algorithms. For the experiments we used three base data sets: (a) 4000 records, and 23 condition attributes; (b) 5000 records, and 20 condition attributes; and (c) 20000 records, and 19 condition attributes. The sets (a) and (b) were randomly generated. The set (c) is based on the set "Letter recognition" from [9]. To the original set we have added three additional columns, each being a combinations of selected columns from the original set (so that more reducts should appear in the results). For each of the databases we have prepared a number of sets of data - 5 sets for (a), 6 sets for (b) and 11 sets for (c). Every set of data was prepared by a random selection of objects from the base sets. For each series of data sets we have performed one experiment for the sequential algorithm, and additionally, 3 experiments - one for each way of decomposition. Below we present the results of the experiment. Tables 5-7 contain the execution times for the sequential version of the algorithm for each of the 3 testing data respectively. In these tables the column 2 shows the total execution time, the columns 3 shows the execution time of computing IND matrix and reduced discernibility function. It is not possible to split times for processing IND matrix and *discernibility function* without the loss of efficiency.

Let us note that computing of the *IND* matrix and *discernibility function* for the first case (Table 5) takes less than 1% of the total processing time. In the 2^{nd} (Table 6) case the computing of *IND* is about 50% of the total time of processing. The number of clauses in prime implicant is smaller for this data set. In the 3^{rd} case (table 7), the computing of *IND* takes more than 99% of the total computing time. Let us note that only for this case the decomposition of *DT* can be justified.

Now we present Tables 8-10. In each table the results of 3 distributed algorithms are presented for each data set respectively.

From Table 8 we can see that for the datasets where *discernibility function* is long and we expect many results, it is better to use vertical decomposition. The vertical decomposition has two advantages: (a) we decompose the phase that

Table 5. Time of computing for sequential method, data set 1

Size (records)	*Total time (ms)*	*IND matrix time (ms)*	*IND* matrix size	*Reducts number*
2000	2483422	29344	513	5131
2500	2144390	41766	475	4445
3000	2587125	60766	555	5142
3500	3137750	80532	532	4810
4000	191390	100266	116	1083

Table 6. Time of computing for sequential method, data set 2

Size (records)	*Total time (ms)*	*IND matrix time (ms)*	*IND* matrix size	*Reducts number*
2500	70735	31457	77	202
3000	68140	46016	41	107
3500	79234	61078	33	72
4000	99500	77407	37	109
4500	127015	100235	42	127
5000	151235	120094	46	131

Table 7. Time of computing for sequential method, data set 3

Size (records)	*Total time (ms)*	*IND matrix time (ms)*	*IND matrix size*	*Reducts number*
11000	668063	650641	16	5
12000	798906	780391	16	5
13000	936375	916641	16	5
14000	1086360	1065375	16	5
15000	1245188	1223016	16	5
16000	1413032	1389782	16	5
17000	1597250	1572843	16	5
18000	1787578	1762015	16	5
19000	1993640	1966718	16	5
20000	2266016	2238078	16	5

Table 8. Parallel methods for date set 1

Size (records)	*A kind of decomposition*									
	DT		*DISC FUNCTION*				*CANDIDATE REDUCTS*			
	S_3	E_3	S_2	E_2	S_3	E_3	S_2	E_2	S_3	E_3
2000	0,41	0,14	3.01	1.5	2.53	0.84	4.33	2.16	7.66	2.55
2500	0,75	0,25	3.03	1.51	2.46	0.82	3.91	1.95	6.64	2.21
3000	0,53	0,18	2.11	1.05	2.11	0.70	3.34	1.67	6.08	2.02
3500	0,70	0,24	3.6	1.8	2.8	0.93	3.55	1.77	6.90	2.30
4000	0,46	0,15	1.57	0.78	1.57	0.52	0.64	0.32	1.0	0.33

Table 9. Parallel methods for date set 2

Size (records)	*A kind of decomposition*									
	DT		*DISC FUNCTION*				*CANDIDATE REDUCTS*			
	S_3	E_3	S_2	E_2	S_3	E_3	S_2	E_2	S_3	E_3
2500	0,72	0,24	1.86	0.93	1.91	0.64	0.79	0.39	0.88	0.29
3000	0,69	0,23	1.36	0.68	1.37	0.46	0.84	0.42	0.84	0.28
3500	0,54	0,18	1.18	0.59	1.22	0.40	0.93	0.47	0.71	0.23
4000	0,73	0,24	1.19	0.60	1.16	0.39	0.96	0.48	0.97	0.32
4500	1,02	0,34	1.20	0.60	1.08	0.36	0.84	0.42	0.90	0.30
5000	0,99	0,33	1.18	0.59	1.10	0.37	0.98	0.49	1.06	0.35

Table 10. Parallel methods for date set 3

Size (records)	*A kind of decomposition*									
	DT		*DISC FUNCTION*				*CANDIDATE REDUCTS*			
	S_3	E_3	S_2	E_2	S_3	E_3	S_2	E_2	S_3	E_3
11000	1.57	0.52	0.99	0.49	1.00	0.33	0.99	0.49	1.00	0.33
12000	1.55	0.52	1.00	0.50	1.00	0.33	1.00	0.50	1.00	0.33
13000	1.58	0.53	1.00	0.50	1.00	0.33	1.00	0.5	1.00	0.33
14000	1.59	0.53	1.00	0.50	0.99	0.33	1.00	0.5	0.99	0.33
15000	1.59	0.53	0.99	0.49	0.99	0.33	0.99	0.49	0.99	0.33
16000	1.60	0.53	1.00	0.50	1.00	0.33	1.00	0.5	1.00	0.33
17000	1.61	0.54	1.00	0.50	1.00	0.33	1.00	0.5	1.00	0.33
18000	1.63	0.54	1.00	0.50	0.99	0.33	1.00	0.5	0.99	0.33
19000	1.64	0.54	1.00	0.50	1.00	0.33	1.00	0.5	1.00	0.33
20000	1.73	0.58	1.00	0.50	0.50	0.33	1.00	0.5	1.00	0.33

takes majority of the time; and (b) joining partial results is less time consuming than merging. For the methods with horizontal decomposition the time of computing depends on the time of merging partial results. By adding another processor not necessarily we get better results - although the conversion to *DNF* is faster, the merging of three sets is more complicated.

In the second case (Table 9) only the method with *discernibility function* decomposition gives good results. Splitting *candidate reducts* was not effective, because conversion from *CNF* to *DNF* takes less than 50% of the total processing time, so the decomposition was made too late. Also splitting DT was not effective, as this method may cause a redundancy in partial results. The best method here is splitting *discernibility function*. It also may cause redundancy in the partial results, but much less then the DT decomposition.

In Table 10 we have an unusual case, because of big number of objects and small number of attributes. The processing of IND takes more than 99% of total time, so we can expect that only the decomposition of DT can give us satisfactory results.

5 Conclusions and Future Work

We have investigated possibilities of decomposing the process of computing the reducts. Three points where the decomposition is feasible have been identified. Based on this, three algorithms of parallel computing of the reducts have been presented and tested. The performed experiments have shown that each of the algorithms has its own specific kind of data sets, for which it is the best. It is therefore an important task to identify at the beginning of the computations which way of paralleling the reduct computations is the most appropriate.

We also expect that for some kind of data combining the three methods can also bring positive results. Special heuristics have to be prepared in order to decide (perhaps dynamically, during the computations) on when and how split the computations. This is the subject of our future research.

References

1. Pawlak, Z.: Rough sets: Theoretical aspects of reasoning about data. Kluwer, Dordrecht (1991)
2. Bazan, J., Nguyen, H.S., Nguyen, S.H., Synak, P., Wróblewski, J.: Rough set algorithms in classification problem. In: Polkowski, L., Tsumoto, S., Lin, T. (eds.) Rough Set Methods and Applications, pp. 49–88. Springer, Physica-Verlag, Heidelberg (2000)
3. Wróblewski, J.: A parallel algorithm for knowledge discovery system. In: PARELEC 1998, pp. 228–230. The Press Syndicate of the Technical University of Bialystok (1998)
4. Wróblewski, J.: Adaptacyjne Metody Klasyfikacji Obiektów. Ph.D thesis, Uniwersytet Warszawski, Wydziaş Matematyki, Informatyki i Mechaniki (2001)
5. Bakar, A.A., Sulaiman, M., Othman, M., Selamat, M.: Finding minimal reduct with binary integer programming in datamining. In: Proc. of the IEEE TENCON 2000, vol. 3, pp. 141–146 (2000)
6. Skowron, A., Rauszer, C.: The discernibility matrices and functions in information systems. In: Slowinski, R. (ed.) Decision Support: Handbook of Applications and Advances of Rough Sets Theory, pp. 331–362. Kluwer, Dordrecht (1992)
7. Susmaga, R.: Parallel computation of reducts. In: Polkowski, L., Skowron, A. (eds.) Rough Sets and Current Trends in Computing, pp. 450–457. Springer, Heidelberg (1998)
8. Karbowski, A., Niewiadomska-Szymkiewicz, E. (eds.): Obliczenia równolegşe i rozproszone. Oficyna Wydawnicza Politechniki Warszawskiej (in Polish) (2001)
9. Asuncion, A., Newman, D.: UCI machine learning repository (2007), `http://www.ics.uci.edu/~mlearn/MLRepository.html`

From Information System to Decision Support System

Alicja Wakulicz-Deja and Agnieszka Nowak

Institute of Computer Science, University of Silesia
Będzińska 39, 41–200 Sosnowiec, Poland
{wakulicz,nowak}@us.edu.pl

Abstract. In the paper we present the definition of *Pawlak's* model of an information system. The model covers information systems with history, systems with the decomposition of objects or attributes and dynamical information systems. Information systems are closely related to rough set theory and decision support systems. The aim of the paper is to characterize the stimulated by Professor *Pawlak* research of the group in Silesian University in information retrieval based on different information systems and in decision support based on rough sets, and to outline the current research projects of this group on modern decision systems.

Keywords: information system, decision support system, rough set theory, clustering methods.

1 Introduction

Information systems and decision support systems are strongly related. The paper shows that we can treat a decision system as an information system of some objects, for which we have the information about their classification. Recently, not so many attention is paid for a classification of information systems in the literature. We deal with a problem of classification based on changes of information systems in the time, what leads in natural way to a concept of dynamic systems. Data analysis in a given information system is possible thanks to defining: the decomposition of system (done on the set of attributes or objects), dependent and independent attributes in data (to remove the attributes that are dependent), whether the attributes or even objects are equivalent, comparison of the objects, attributes and even the whole systems. The paper also presents that the model of information system created by Professor *Pawlak* is very useful for retrieving information. One of the different methods of retrieving information, so called *atomic components method*, was proposed by Professor *Pawlak*, and it is presented in the paper with all basic assumptions. The relation between information systems and rough set theory with decision support systems, where researches are concerned with the classificatory analysis of imprecise, uncertain or incomplete information or knowledge expressed in terms of data acquired from experience, is also presented in the paper. It also consider the methods of reduction the set of attributes and rule induction method's that have been applied to knowledge discovery in databases, whose empirical results obtained show that they are very powerful and that some important knowledge has been extracted from databases. Because of that, in the paper, the results of the stages of different researches that were done (i.e. diagnosis support system used in child neurology and it is a notable example of a complex multistage diagnosis process) and all of the researches that are planed to do at the Silesian University, are presented. It is supposed to explain Professor *Pawlak's* invaluable contribution to the domain of information and decision support systems.

The notion of an information system formulated by Professor *Pawlak* and developed with his co-workers, is now a well developed branch of data analysis formalisms. It is strongly related to (but different from) the relational database theory on the one hand and to fuzzy set theory on the other. In this paper we consider the connection between the theory of information and information retrieval systems with rough set theory and decision support systems. It is obvious that model of a system created by Professor *Pawlak* makes data description and analysis simple and very reliable.

2 Information System

An information system consists of a set of objects and attributes defined on this set. In information systems with a finite number of attributes, there are classes created by these attributes (for each class, the values of the attributes are constant on elements from the class).

Any collection of data, specified as a structure:

$$S = \langle X, A, V, q \rangle$$

such that X is a non-empty set of objects, A is a non-empty set of attributes, V is a non-empty set of attributes' values: $V = \bigcup_{a \in A} V_a$ and q is an *information function* of $X \times A \to V$, is referred to as an information system. The set $\{q(x,a) : a \in A\}$ is called *information about the object* x or, in short, a record of x or a row determined by x.

Each attribute a is viewed as a mapping $a : X \to V_a$ which assigns a value $a(x) \in V_a$ to every object x. A pair (a, v), where $a \in A$, and $v \in V_a$, is called a *descriptor.*

In information systems, the *descriptor language* is a formal language commonly used to express and describe properties of objects and concepts.

More formally, an information system is a pair $\mathcal{A} = (U, A)$ where U is a non-empty finite set of objects called the *universe* and A is a non-empty finite set of attributes such that $a : U \to V_a$ for every $a \in A$. The set V_a is called the *value set* of a. Now we will discuss which sets of objects can be expressed (defined) by formulas constructed by using attributes and their values. The simplest formulas d, called *descriptors*, have the form (a, v) where $a \in A$ and $v \in V_a$.

In each information system S the information language $L_S = \langle AL, G \rangle$ is defined, where AL is the alphabet and G is the grammar part of that language.

AL is simply a set of all symbols which can be used to describe the information in such a system, e.g.:

1. $\{0, 1\}$ (constant symbols),
2. A - the set of all attributes,
3. V - a set of all the values of the attributes,
4. symbols of logical operations like $\tilde{}$, $+$ and $*$,
5. and naturally brackets, which are required to represent more complex information.

G - the grammar part of the language L_S defines syntax with T_S as the set of all possible forms of terms (a *term* is a unit of information in S) and its meaning (semantics). A simple descriptor $(a, v) \in T_S(a \in A$ where $v \in V_a)$. If we denote such a descriptor (a, v) as the term t, then following term formations will be also possible: $\neg t, t + t^{'}, t * t^{'}$, where $t, t^{'} \in T_S$. The meaning is defined as a function σ which maps the set of terms in a system S in a set of objects X, $\sigma : T_S \to P(x)$, where $P(x)$ is the set of the subsets of X. The value of σ for a given descriptor (a, v) is defined as following [49]:

1. $\sigma(a, v) = \{x \in X, q_x(a) = v\}$,
2. $\sigma(\neg t) = X \setminus \sigma(t)$,
3. $\sigma(t + t^{'}) = \sigma(t) \cup \sigma(t^{'})$ and
4. $\sigma(t * t^{'}) = \sigma(t) \cap \sigma(t^{'})$.

2.1 Information Table

Information systems are often represented in a form of tables with the first column containing objects and the remaining columns, separated by vertical lines, containing values of attributes. Such tables are called *information tables* (an example is presented in Table 1).

The definition of this system is as follows:

$$S = \langle X, A, V, q \rangle,$$

where $X = \{x_1, \ldots, x_8\}$, $A = \{a, b, c\}$, $V = V_a \cup V_b \cup V_c$, $V_a = \{a_1, a_2\}$, $V_b = \{b_1, b_2\}$, $V_c = \{c_1, c_2, c_3, c_4\}$ and $q : X \times A \to V$.

For instance, $q(x_1, a) = a_1$ and $q(x_3, b) = b_1$.

Table 1. An information system - an information table

student	a	b	c
x_1	a_1	b_1	c_1
x_2	a_1	b_1	c_2
x_3	a_2	b_1	c_3
x_4	a_2	b_1	c_4
x_5	a_1	b_2	c_1
x_6	a_1	b_2	c_2
x_7	a_2	b_2	c_3
x_8	a_2	b_2	c_4

Before we start considering the properties of an information system, it is necessary to explain what the *information* in such a system means. The *information* in the system S is a function ρ with the arguments on the attributes set A and its values, which belong to the set V ($\rho(a) \in V_a$). As long as the sets of the objects, attributes and their values are finite, we know exactly how many (different) pieces of information in a given system S comprises, and the number is equal to $\bigcap_{a \in A} card(V_a)$. The information ρ assigns a set of the objects X_ρ that $X_\rho = \{x \in X : q_x = \rho\}$. We call them *indiscernible*, because they have the same description. If we assume that $B \subseteq A$ then each subset B of A determines a binary relation $IND_A(B)$, called an *indiscernibility relation.* By the *indiscernibility relation* determined by B, denoted by $IND_A(B)$, we understand the equivalence relation:

$$IND_A(B) = \{\langle x, x^{'} \rangle \in X \times X : \forall_{a \in B}[a(x) = a(x^{'})]\}.$$

For a given information system it is possible to define the comparison of the objects, attributes and even the whole systems. We can find some dependent and independent attributes in data, we can check whether the attributes or even objects are equivalent. An important issue in data analysis is to discover dependencies between attributes. Intuitively, a set of attributes D depends totally on a set of attributes C if the values of attributes from C uniquely determine the values of the attributes from D. If D depends totally on C then $IND_A(C) \subseteq IND_A(D)$. This means that the partition generated by C is finer than the partition generated by D.

Assume that a and b are attributes from the set A in a system S. We say that b depends on a ($a \rightarrow b$), if the indiscernibility relation on a contains in the indiscernibility relation on b: $\widetilde{a} \subseteq \widetilde{b}$. If $\widetilde{a} = \widetilde{b}$ then the attributes are equivalent. The attributes are dependent if any of the conditions: $\widetilde{a} \subseteq \widetilde{b}$ or $\widetilde{b} \subseteq \widetilde{a}$ is satisfied. Two objects $x, y \in X$ are *indiscernible* in a system S relatively to the attribute $a \in A$ ($x_{\widetilde{a}}y$) if and only if $q_x(a) = q_y(a)$. In the presented example, the objects x_1 and x_2 are indiscernible relatively to the attributes a and b. The objects $x, y \in X$ are *indiscernible* in a system S relatively to all of the attributes $a \in A$ ($x_{\widetilde{S}}y$) if and only if $q_x = q_y$. In the example there are no indiscernible objects in the system S. Each information system determines unequivocally a partition of the set of objects, which is some kind of classification. Finding the dependence between attributes let us to reduce the amount of the information which is crucial in systems with a huge numbers of attributes. Defining a system as a set of objects, attributes and their values is necessary to define the algorithm for searching the system and updating the data consisted in it. Moreover, all information retrieval systems are also required to be implemented in this way. The ability to discern between perceived objects is also important for constructing various entities not only to form reducts, but also decision rules and decision algorithms.

2.2 An Application in Information Retrieval Area

The information retrieval issue is the main area of the employment of information systems. An information retrieval system, in which the objects are described by their features (properties), we can define as follows: Let us have a set of objects X and a set of attributes A. These objects can be books, magazines, people, etc. The attributes are used to define the properties of the objects. For the system of books, the attributes can be *author*, *year*, *number of sheets*. An information system which is used for information retrieval should allow to find the answer for a query. There are different methods of retrieving information. Professor *Pawlak* proposed the *atomic components method* [2,49]. Its mathematical foundation was defined in [5] and [6]. This method bases on the assumption that each question can be presented in the normal form, which is the sum of the products with one descriptor of each attribute only. To make the system capable of retrieving information it is required to create the information language (*query language*). This language should permit describing objects and forming user's queries. Naturally enough, such a language has to be universal for both the natural and system language. Owing to this, all steps are done on the language level rather than on the database level. The advantages of information languages are not limited to the aforementioned features. There are a lot of systems that need to divide the information, which is called the decomposition of the system. It allows improving the time efficiency and make the updating process easy, but also enables the organization of the information in the systems. Information systems allow collecting data in a long term. It means that some information changes in time, and because of that, the system has a special property, which is called the dynamics of the system. Matching unstructured, natural-language queries and documents is difficult because both queries and documents (objects) must be represented in a suitable way. Most often, it is a set of terms, where a*term* is a unit of a semantic expression, e.g. a word or a phrase. Before a retrieval process can start, sentences are preprocessed with stemming and removing too frequent words (stopwords). The computational complexity when we move from simpler systems to more compound increases. For example, for atomic component retrieval method, the problem of rapidly growing number of atomic component elements is very important. Assuming that A is a set of attributes, and V_a is a set of values of attribute a, where $a \in A$, in a given system we achieve a $\bigcap_{a\in A} \overline{\overline{V_a}}$ objects to remember. For example, if we have a 10 attributes in a given system S, and each of such attributes has 10 values, we have to remember 10^{10} of elements.

2.3 System with Decomposition

When the system consists of huge set of data it is very difficult in given time to analyse those data. Instead of that, it is better to analyze the smaller pieces (subsets) of data, and at the end of the analysing, connect them to one major system. There are two main method of decomposition: with *attributes* or *objects*. A lot of systems are implemented with such type of decomposition.

System with object's decomposition. If it is possible to decompose the system $S = \langle X, A, V, q\rangle$ in a way that we gain subsystems with smaller number of objects, it means that:

$$S = \bigcup_{i=1}^{n} S_i,$$

where $S_i = \langle X_i, A, V, q_i \rangle$, $X_i \subseteq X$ and $\bigcup_i X_i = X$, $q_i : X_i \times A \to V$, $q_i = q|_{X_i \times A}$.

System with attributes's decomposition. When in system S there are often the same types of queries, about the same group of attributes, it means that such system should be divided to subsystems S_i in a way that:

$$S = \bigcup_{i} S_i,$$

where $S_i = \langle X, A_i, V_i, q_i \rangle$, $A_i \subseteq A$ and $\bigcup_i A_i = A$, $V_i \subseteq V$, $q_i : X \times A_i \to V_i$, $q_i = q|_{X \times A_i}$.

Decomposition lets for optimization of the retrieval information process in the system S. The choice between those two kind of decomposition depends only on the type and main goal of such system.

2.4 Dynamic Information System and System with the History

In the literature information systems are classified according to their purposes: documentational, medical or management information systems. We propose different classification: those with respect to dynamics of systems. Such a classification gives possibility to:

1. Perform a joint analysis of systems belonging to the same class,
2. Distinguish basic mechanisms occuring in each class of systems,
3. Unify design techniques for all systems of a given class,
4. Simplify the teaching of system operation and system design principles.

Analysing the performance of information systems, it is easy to see that the data stored in those systems are subject to changes. Those changes occur in definite moments of time. For example: in a system which contains personal data: age, address, education, the values of these attributes may be changed. Thus time is a parameter determining the state of the system, although it does not appear in the system in an explicit way. There are systems in which data do not change in time, at least during a given period of time. But there are also systems in which changes occur permanently in a determined or quite accidental way.

In order to describe the classification, which we are going to propose, we introduce the notion of a *dynamic* information system, being an extension of the notion of an information system presented by Professor *Pawlak*.

Definition 1. *A dynamic information system is a family of ordered quadruples:*

$$S = \{\langle X_t, A_t, V_t, q_t \rangle\}_{t \in T} \tag{1}$$

where:

– T *- is the discrete set of time moments, denoted by numbers* $0, 1, \dots, N$,
– X_t *- is the set of objects at the moment* $t \in T$,

– A_t - *is the set of attributes at the moment* $t \in T$,
– $V_t(a)$ - *is the set of values of the attribute* $a \in A_t$,
– $V_t := \bigcup_{a \in A_t} V_t(a)$ - *is the set of attribute values at the moment* $t \in T$,
– q_t - *is a function which assigns to each pair* $\langle x, a \rangle$, $x \in X_t$, $a \in A_t$, *an element of the set* V_t, *i.e.* $q_t : X_t \times A_t \to V_t$.

An ordered pair $\langle a, v \rangle$, $a \in A_t$, $v \in V_t(a)$ *is denoted as a descriptor of the attribute* a. *We will denote by* $q_{t,x}$ *a map defined as follows:*

$$q_{x,t} : A_t \to V_t, \tag{2}$$

$$\bigwedge_{a \in A_t} \bigwedge_{x \in X_t} \bigwedge_{t \in T} q_{t,x}(a) := q_t(a, x) \tag{3}$$

Let $Inf(S) = \{V_t^{A_t}\}_{t \in T}$ be a set of all functions from A_t to V_t for all $t \in T$. Functions belonging to $Inf(S)$ will be called informations at instant t, similarly, the functions $q_{t,x}$ will be called the information about object x at instant t in information system S. Therefore, an information about an object x at instant t is nothing else, but a description of object x, in instant t, obtained by means of descriptors. We will examine closer the changes, which particular elements (X,A,V,q) of a dynamic system may undergo in certain time moments (see also [46,47]). Systems, whose all parameters do not depend on time are discussed in [7]. Here we deal with the dynamic systems in which the descriptions of objects depend essentialy on time. It is useful to observe at the begining that any dynamic system belongs to one of two classes of systems: *time-invariant* and *time-varying* system.

Definition 2. *Time-invariant system is the dynamic system such that:*

1. $Z_t := \bigcap_{t \in T} Dq_t \neq \emptyset$ *and*
2. $\bigwedge_{t,t' \in T} \bigwedge_{(x,a) \in Z_T} q_t(x, a) = q_{t'}(x, a)$
where Dq_t - *domain of function* q_t.

Definition 3. *Time-varying system is the dynamic system such that:*

1. $Z_T := \bigcap_{t \in T} Dq_t = \emptyset$ *or*
2. $Z_T \neq \emptyset$ *and* $\bigvee_{t,t' \in T} \bigvee_{(x,a) \in Z_T} q_t(x, a) \neq q_{t'}(x, a)$.

2.5 Time-Invariant Systems

Let X_T, A_T be sets of objects and attributes of the dynamic system defined as follows:

– $X_T := \bigcap_{t \in T} X_t$,
– $A_T := \bigcap_{t \in T} A_t$.

It is evident by the definition of the *time-invariant* system that a dynamic system is time-invariant if and only if

$$q := q_t|_{Z_T} \tag{4}$$

does not depend on t and $Z_T = X_T \times A_T$.

It means that any *time-invariant* system $S = \{< X_t, A_t, V_t, q_t >\}_{t \in T}$, has a subsystem $S^{'}$ in *Pawlak*'s notion which is time independant

$$S^{'} =< X_T, A_T, q(Z_T), q > .$$

Let us consider a system of library information in which books are objects, the set of attributes is given by author's name, title, publisher's name, year of issue, subject, etc. and attributes values are given in natural language [48,49].

Let us consider time evolution of this system on the example given by its subsystem connected with four books:

- b_1 = *C.J.Date, An Introduction to database systems.*
- b_2 = *G.T.Lancaster, Programming in COBOL.*
- b_3 = *Ch.T.Meadow, The analysis of Information Systems.*
- b_4 = *G.Salton, The SMART retrieval system.*

and four attributes: *publisher, year of issue, number of pages, subject.* The history of our library in years 1980, 1981, 1982 described by our subsystem depends on two events. Begining from 1981 out library information was enriched with the information about subject of book and the book b_4 was bought, and in 1982 the book b_3 was lost. This situation is given by dynamic system $S = \{< X_t, A_t, V_t, q_t >\}_{t=1980,1981,1982}$, described in the tables 2, 3, and 4. Table 5 presents a time-invariant subsystem $S^{'} = \{< X_t, A_t, V_t, q >\}$.

It is easy to see that in the dynamic system described above $X_T = \{b_1, b_2\}$, A_T=$\{Publisher, Year, Pages\}$, and V_T is given below what propes that $q|_{X_T \times A_T}$ is time independent i.e. the system described in the example is time-invariant.

Table 2. $S = \{< X_t, A_t, V_t, q_t >\}_{t=1980}$

$X_{1980} \backslash A_{1980}$	Publisher	Year	Pages
b_1	Addison-Wesley Publish. Comp.Inc.,USA	1977	493
b_2	Pergamon Press, Oxford, New York	1972	180
b_3	John Wiley & Sons Inc., New York	1967	339

Table 3. $S = \{< X_t, A_t, V_t, q_t >\}_{t=1981}$

$X_{1981} \backslash A_{1981}$	Publisher	Year	Pages	Subject
b_1	Addison-Wesley Publish. Comp.Inc.,USA	1977	493	Databases
b_2	Pergamon Press, Oxford, New York	1972	180	Programming
b_3	John Wiley & Sons Inc., New York	1967	339	Information Sys.
b_4	Prentice-Hall Inc., Englewood-Cliffs, USA	1971	585	Retrieval Sys.

Table 4. $S = \{< X_t, A_t, V_t, q_t >\}_{t=1982}$

$X_{1982} \backslash A_{1982}$	Publisher	Year	Pages	Subject
b_1	Addison-Wesley Publish. Comp.Inc.,USA	1977	493	Databases
b_2	Pergamon Press, Oxford, New York	1972	180	Programming
b_4	Prentice-Hall Inc., Englewood-Cliffs, USA	1971	585	Retrieval Systems

Table 5. Time-invariant subsystem $S^{'} = \{< X_t, A_t, V_t, q >\}$

$X_T \backslash A_T$	Publisher	Year	Pages
b_1	Addison-Wesley Publish. Comp.Inc.,USA	1977	493
b_2	Pergamon Press, Oxford, New York	1972	180

2.6 Time-Varying Systems

If $\bigcap_{t \in T} X_t = \emptyset$ or $\bigcap_{t \in T} A_t = \emptyset$ i.e. $Z_T = \emptyset$ then the system is obviously time dependent on T since there does not exist an element x beloniging to all X_t or a belonging to all A_t.

If $Z_T \neq \emptyset$ then the dynamic system $S = \{< X_t, A_t, V_t, q_t >\}_{t \in T}$ has a subsystem

$$S^{'} = \{< X_T, A_T, q_t(Z_T), q_t|_{Z_T} >\},$$

$t \in T$ and we can observe that this system is not time-invariant, since by the definition of the time-varying system there exist $t, t^{'} \in T$ and $(x, a) \in Z_T$ that

$$q_t(x, a) \neq q_{t^{'}}(x, a).$$

A system which contains information about students [27] is good example of a system with *time-varying* information. The set of objects is the set of all students of a fixed University [*Faculty,Course*]. As a set of attributes we may choose, for example: *STUDY-YEAR, GROUP, MARK-OF-MATH, MARK-OF-PHYSICS,AV-MARK* and so on. Descriptors are as before, pairs of the form <*attribute, attribute value*>, where the sets of attribute values are as follows:

$$||STUDY-YEAR|| = \{I, II, III, \ldots\},$$
$$||GROUP|| = \{1, 2, 3, \ldots\},$$
$$||MARK-OF-MATH|| = \{2, 3, 4, 5\},$$
$$||MARK-OF-PHYSICS|| = \{2, 3, 4, 5\},$$
$$||AVERAGE\ MARK|| = \{2, 2.1, 2.2, \ldots, 5\}.$$

Let us assume that student changes the study year if his average mark lays between 3 and 5. If not the student ramains on the same year of studies. If there is not a change in the study year the student can change the students group. Let us consider the history of three students s_1, s_2, s_3 begining with the first year of their studies during the following three years. The situation in the system

Table 6. First year of observation

$X_1 \backslash A_1$	Year	Group	Av.mark
s_1	I	1	–
s_2	I	1	–
s_3	I	2	–

Table 7. Second year of observation

$X_2 \backslash A_2$	Year	Group	Av.mark
s_1	I	3	3.1
s_2	II	1	4.1
s_3	II	2	3.3

Table 8. Third year of observation

$X_3 \backslash A_3$	Year	Group	Av.mark
s_1	II	3	3.1
s_2	III	1	4.8
s_3	II	1	3.7

is described in tables 6, 7, and 8. One can observe that $X_T = \{s_1, s_2, s_3\}$, $A_T = \{STUDY - YEAR, GROUP, AV - MARK\}$, and

$$q_t(s_1, STUDY\_YEAR) = \begin{cases} I \quad t = 1 \text{ year of observation} \\ I \quad t = 2 \text{ year of observation} \\ II \; t = 3 \text{ year of observation} \end{cases} \tag{5}$$

what means that the system is the time-varying system.

2.7 Variability of Information in Dynamic Systems

In time-varying systems we can observe various types of information changes. If the set $Z_T = \{ \bigcap_{t \in T} X_T \} \cap \{ \bigcap_{t \in T} A_T\} \neq \emptyset$ then the important features of the character of changes of the information in time is described by the dynamic subsystem $S^{'}$.

$$S^{'} = \{< X_T, A_T, q_t(Z_T), q_t|_{Z_T} >\}_{t \in T}.$$

In the subclass of dynamic systems, represented by system $S^{'}$, the state of the system depends on time t by the family $\{q_t\}_{t \in T}$ only.

Due to a way of realization of this subclass of systems in the practise it is sensible to consider such a realization of systems, which allows to determine values of the function:

$$f(x, a, q_{t-1}(x, a), \ldots, q_{t-i}(x, a))$$

for all $x \in X_T$, $a \in A_T$ and $t \in T$.

By f we denote any function which is feasible in the considered realization, and by i we denote so called depth of information and can assume values $0, 1, \ldots, I$. When $i = 0$ the function f depends on x and a only. One can observe that such realizations of systems are not giving possibility of determining values of a function which explicitly depends on t. This is one of features which distinguish dynamic information systems from data processing systems. From the point of view of the realizations described above, any dynamic system belongs to the one of the subsequent classes:

1. Systems with determined variability SDV.

   A dynamic system belongs to SDV if and only if:
   – $\bigwedge_{(x,a)\in Z_T}$ there exist initial values $q_{-1}(x,a), \ldots, q_{-i}(x,a) \in \bigcup_{t\in T} V_t$ such that:
   $\bigwedge_{t\in T} \bigwedge_{(x,a)\in Z_T} q_t(x,a) = f(x, a, q_{t-1}(x,a), \ldots, q_{t-i}(x,a))$
   for properly choosen (feasible) function f.

2. Systems with predictable variability SPV.

   A dynamic system belongs to SPV if and only if:
   – it does not belong to SDV,
   – there exist $T_1, \ldots, T_M \subset T$ ($\bigcup_{j=1}^{M} T_j = T$, $T_j \cap T_k = \emptyset$, for $j \neq k$, $j, k = 1, \ldots, M$, card $T_j > 1$ for $j = 1, \ldots, M$), and feasible functions $f_1, \ldots, f_M$ such that:
   $\bigwedge_{t\in T_j} \bigwedge_{(x,a)\in Z_T} q_t(x,a) = f_j(x, a, q_{t-1}(x,a), \ldots, q_{t-j}(x,a))$
   for properly choosen initial values $q_{-1}(x,a), \ldots, q_{-ij}(x,a)$.

3. Systems with unpredictable variability SUV.

   A dynamic system belongs to SUV if and only if:
   – it does not belong to SDV or SPV.
   It is worthy to underline that to SUV can belong systems whose structure is formally simple. For example, the system whose information function is determined as follows:

$$q_t(x,a) = \begin{cases} f_1(x, a, q_{t-1}(x,a), \ldots, q_{t-i_1}(x,a)) \text{ or} \\ f_2(x, a, q_{t-1}(x,a), \ldots, q_{t-i_2}(x,a)) \end{cases} \tag{6}$$

   belongs to SUV as long as there is not determined for which t: f_1 and for which f_2 is applied.

2.8 Examples of Time-Varying Systems

Examples of systems belonging to SDV, SPV and SUV classes are given here. An example of a system with determined variability SDV can be a system of patient supervision (medical information). The objects of this system are

Table 9. The prescribitions of medicaments and tests for patients

$X\backslash A$	Test blood	X-ray lungs	Peniciline injections	Vitamins
p_1	1	-	-	0.2 C
p_2	-	-	1	-
p_3	-	+	-	0.03 B_1
p_4	1	-	-	-

Table 10. The physician prescription for a given patient

$A\backslash t$	0	1	2	3	4	5	6	7	8	9	10
P	1	1	1	1	1	1	0	0	0	0	0
T	0	0	1	0	0	1	0	0	1	0	0

patients. The attributes are, for example, test of blood morphology, lungs X-ray, prescribed penicillin, prescribed doses of vitamins (in mg), etc. The following table (Table 9) presents prescribitions of medicaments and tests for patients p_1, p_2, p_3, p_4 at the begining of the considered system performance ($t = 0$). Let us describe the system performance on the example of the patient p_2 who after small surgery get an bacterial infection. Physician prescription is as follows: Penicillin injections P for six forthcoming days, blood morphology test T every thrid day. This prescription gives the following table (Table 10) of function $q_t(p_2, P)$ and $q_t(p_2, T)$. One can observe that using the Boolean algebra notion these functions can be written in the following form

$$* \begin{cases} q_t(p_2, P) = q_{t-1}(p_2, P)[\overline{q_{t-2}}(p_2, P) + \overline{q_{t-7}}(p_2, P)] \\ q_t(p_2, T) = \overline{q_{t-1}}(p_2, T).\overline{q_{t-2}}(p_2, T) \end{cases} \tag{7}$$

if only initial values are given as follows

$$** \begin{cases} q_{-1}(p_2, P) = 1 \\ q_{-j}(p_2, P) = 0 \text{ for } j = 2, 3, \ldots, 7 \text{ information depth} = 7, \\ q_{-k}(p_2, T) = 1 \text{ for } k = 1, 2 \text{ information depth} = 2. \end{cases} \tag{8}$$

The formulas $/*/$, $/**/$ convince us that described system (at least reduced to object p_2 and attributes P and T) is SDV. Other systems of the class SDV can be found in [29,28,30]. As a example of SPV we use the system with students, and we assume that T_1, T_2, T_3 are time intervals determined as follows:

T_1: from Oct.1st 1980 to Sept.30th 1981,
T_2: from Oct.1st 1981 to Sept.30th 1982,
T_3: from Oct.1st 1982 to Sept.30th 1983.

It is easy to see that the function $q_t(s_i, Y)$, $q_t(s_i, G)$, $q_t(s_i, A.m)$, $i = 1, 2, 3.$ are constants in each time interval T_1, T_2, T_3. Therefore on each interval T_1, T_2, T_3 this function are realizable (information depth $= 0$) and the system belongs to

Table 11. The example of the system belonging to the *SUV*

$X\backslash A$	Storage	Prod.division (I)	Prod.division(II)
M_1	200	50	30
M_2	100	10	20
M_3	0	5	4

SPV. Finally, let us consider a system which describes materials managment in a factory. Objects in this system are different types od materials. The attributes can be here production divisions and /or workstands and the main storage. The attribute values are given in units (of weight, measure, etc.) which are natural for the described object. Let us consider the system of this kind reduced to three objects a storage and two production divisions as attributes. Let the attribut values be given in the *Table* 11. It is obvious that a state of resources of the objects M_i on the production division K (K = I,II) depends not only of information function $q_{t-1}(M_i, K)$ but also on information function defined on other attributes i.e. depends on $q_{t-1}(M_1, II)$, $q_{t-1}(M_1, St.)$, $q_{t-1}(M_1, I)$ therefore it is not a function which can be used as information function due to the definition of dynamic system. Moreover the values of functions $q_t(M_i, St.)$ are not determined a priori and generally we can not determine the moments in which these values will be changed. This system of course does not belong to SDV or SPV. Therefore it belongs to SUV. Examples of systems belonging to the SUV class can be found in [27,31] also.

2.9 Influence of Foundations of a System on Its Classification

Analysing foundations of a real system we can determine to which of described above classes the system belongs. Thus e.g., if we assume, that objects of the system are documents with static or rarely changing descrptions, then this system will belong to the class of invariant systems. The characteristics of most library systems imply directly their belonging to the class of time-invariant systems. In the same way, the assumption about variability in documents descriptions will suggest, that a system containing such documents belongs to the class of systems with time-varying information. Of course, if we are able to determine moments in which descrptions changes will occur, then it will be the system with predictable variability (SPV). If we are not able to determine these moments - we will obtain a system with unpredictable variability (SUV). Some systems are a priori classified as systems with determined variability (SDV), because the knowledge of “histories” of objects is one of the requirements, as in medical systems for example. So, foundations of the realized information system decide a priori about its classification - which, in consequence, suggests a priori certain performance mechanisms of this system. Many of existing systems are actually packages of systems belonging to different classes (e.g. medical system may consist with a module of registration which is an time-invariant system and with a module of patients supervison which belongs to the class of time-varying systems). In this case every of modules is designed as a system of an appropriate class. The classification resulted from the analysis of performance of information systems, can be a convenient tool for designing purposes. When somebody starts system designing, he has a good knowledge of systems foundation and parameters but generally he can not predict proper mechanisms of system performance. In this situation, as was stated above, he can determine a class to which the system belongs. This allows him to choose adequate mechanisms of system performance.

2.10 Performance Mechanisms in Dynamic Systems

In a realization of information systems we should made decisions about a structure of database and the way of its updating, on the retrieval method and a retrieval language we are going to use and on the mode of operation which will be used in the system. In the forthcoming we give some remarks how these decision depend on the fact that the considered system belongs to one of the determined classes i.e. class of Invariant Systems, class of Systems with Determined Variability (SDV), class of Systems with Predictable Variability (SPV), class of Systems with Unpredictable Variability (SUV).

Database and its updating
At first let us consider invariant systems. The database of the invariant system is static throughout the period of performance. A reorganization of the database, if desired, is realized after the period of performance and consists in creating a new database. In systems with time-varying information the database changes during the action of the system. In systems with determined variability (SDV) we have to store information about an object in the past because this information is necessary for determining the actual information about this object. Thus the "history", with prescribed depth of information about objects, should be stored in the database. In systems with predictable variability (SPV) actualization and reorganization of the database ought to be executed at certain moments, in which changes are predicted. These are mainly changes in descriptions of objects. The database reorganization (actualization) does not necessarily involve changes in programs operating on the database. In systems with unpredictable variability (SUV) any execution of the retrieval process ought to be preceded by the actualization of descriptions of objects. In all systems with time-varying information we can have at the same period an actualization of the set od objects, set of attributes and set of descriptors, as in invariant systems.

Retrieval method and information retrieval language
Because of the specyfic character of the database and actualization process, one prefers the exhaustice search as a retrieval method for invariant systems.In such a case an extension of database does not results in the retireval method. At most, in order to speed up the system performance, one may apply the methods of inverted files or linked lists. These methods are more useful for some systems with predictable variability (information depth =0). There, when the system action is stoped, the database can be actualized along with inverted files or linked lists updating. In these systems there is no need for developing special information

retrieval languagees, because languages based on thesauruses, indexing or decimal classification seem to be sufficiently efficient. However in the invariant systems and systems with predictable variability one can prefer a specific method of retrieval. For a realization of the systems with time-varying information a grouping of informations and random access to descriptions of these groups or an individual description of object is essential. Mathematical methods of retrieval seem to be the most convenient in this case (for example: *Lum*'s methods [23] or the *atomic component method* with decomposition of system). These retireval algorithms allow us to find quickly a particular information, they also simplify the updating process. In the case of systems with determined variability (SDV), this problem looks a bit different, because a new information is constantly created and has to be stored. In this case the method of linked lists seem to be as good as mathematical methods (e.g. the method of atomic components). In the method of linked lists an actual information about the object is obtained by consideration of a chain of a determined length given by the depth of information. In the systems with time-varying information a language based on descriptors is the most convenient, for information retrieval, since it allows us easy to write/read informations described by means of codes which are equivalents of descriptors. Moreover in this case the descriptions of objects are determined by the values of attributes. Informations in time-varynig systems are always described by means of codes, therefore all output informations are translated onto the natural language. Consequently, from the user's point of view, there is no difference if the system uses the descriptor or another language. In some cases, when this translation can be omitted (e.g. in medical systems, which are used by medical servise) the descriptors ought to be introduced in accordance with codes accepted by a user. Here we ought to mention interactive languages, which seem to be necessary for most systems with time-varying information (the necessity of a dialogue with the system) but they will be discussed latter on, along with the operation mode of dynamic systems.

Operation mode

Let us consider now the continuous operation mode and the batch operation mode in an information system. The continuous operation mode consists in current (i.e. $\forall_{t\in T}$) information feeding, therefore we hace current data base updating. This operation mode will occur in systems with unpredictable variability, an actualization processes will be executed in turns with retrieval processes. In most cases, however, information systems work in batch operation mode, which means that in certain moments actualization and reorganization processes take place. This operation mode can be used in invariant and time-varying systems with predictable variability (SPV). A case with the interactive operation mode is a bit different, since a user is able to communicate with a system. If this mode is used only for rerieval purposes (to find more complete or relevant information), then it can be applied to a system of arbitrary class. But if goal of this dialogue is to create a new data base structure (internal changes), then interactive systems are limited down to the class of systems with unpredictable variability (SUV). At the end let us mention that due to the structure of dynamic model discussed here (definition of the dynamic information system) performance mechanisms are applied to any pair (x, a), $x \in X_T$, $a \in A_T$ separately. This all reorganizations of the model which are based on concurrent processing and multi-access give high efficiency of the information system in the practise.

Conclusion
In this paper a possibility of introducing dynamics in *Pawlak*'s model of systems is presented. In the most situations of practise this model is more convenient then the classical (relational) model. It is due to the fact that in Pawlak's model, information about an object are given by functions, while in the classical model informations are determined by relations. This simplifies a description of systems and their analysis, which is important not only for system designing but also for teaching of system operation. Authors think, that the only way of teaching how to use the system and how to design it, goes by understanding of system operation mechanisms. For the model presented here the proposed classification allows to fullfil this goal easier. The model of information system created by *Pawlak* is very useful to built and analysis in different types of retrieval information systems. The document information systems are very specific type of information systems and *Pawlak*'s model is very good to define the informations in it.

3 Decision Support Systems

When data mining first appeared, several disciplines related to data analysis, like statistics or artificial intelligence were combined towards a new topic: extracting significant patterns from data. The original data sources were small datasets and, therefore, traditional machine learning techniques were the most common tools for this tasks. As the volume of data grows these traditional methods were reviewed and extended with the knowledge from experts working on the field of data management and databases. Because of that, information systems with some data-mining methods start to be the decision support systems. Decision support system is a kind of information system, which classifies each object to some class denoted by one of the attributes, called *decision* attribute.

While the information system is simply a pair of the form U and A, the *decision support system* is also a pair $S = (U, C \cup \{d\})$ with distinguished attribute d. In case of decision table the attributes belonging to C are called *conditional attributes* or simply *conditions* while d is called *decision*.

We will further assume that the set of decision values is finite.

The i-th *decision class* is a set of objects:

$$C_i = \{x \in U : d(x) = d_i\},$$

where d_i is the i-th decision value taken from decision value set $V_d = \{d_1, \ldots, d_{|V_d|}\}$. Let us consider the decision table presented as Table 12. In presented system (with informations about students):

$C = \{a, b, c\}$,
$D = \{d\}$.

Table 12. Decision table

student	a	b	c	d
x_1	a_1	b_1	c_1	T
x_2	a_1	b_1	c_2	T
x_3	a_2	b_1	c_3	T
x_4	a_2	b_1	c_4	N
x_5	a_1	b_2	c_1	N
x_6	a_1	b_2	c_2	T
x_7	a_2	b_2	c_3	T
x_8	a_2	b_2	c_4	N

Having indiscernibility relation we may define the notion of reduct. In case of decision tables *decision reduct* is a set $B \subset C$ of attributes, which cannot be further reduced and $IND(B) \subseteq IND(d)$.

Decision rule is a formula of the form:

$$(a_{i_1} = v_1) \wedge \ldots \wedge (a_{i_k} = v_k) \Rightarrow (d = v_d),$$

where $1 \leq i_1 < \ldots < i_k \leq m, v_j \in V_{a_ij}$.

We can simply interpret such formula similar to natural language with *if* and *then* elements.

In given decision table the decision rule for object x_1 is given as:

$$\textit{if } (a = a_1) \textit{ and } (b = b_1) \textit{ and } (c = c_1) \textit{ then } (d = T),$$

the same as

$$(a = a_1) \wedge (b = b_1) \wedge (c = c_1) \rightarrow (d = T).$$

Atomic subformulas $(a_{i_1} = v_1)$ are called *conditions, premises*. We say that rule r is applicable to object, or alternatively, the object matches rule, if its attribute values satisfy the premise of the rule.

Each object x in a decision table determines a decision rule:

$$\forall_{a \in C}(a = a(x)) \Rightarrow (d = d(x))),$$

where C is set of conditional attributes and d is decision attribute. Decision rules corresponding to some objects can have the same condition parts but different decision parts. We use decision rules to classify given information. When the information is uncertain or just incomplete there is need to use some additional techniques for information systems. Numerous methods based on the rough set approach combined with Boolean reasoning techniques have been developed for decision rule generation.

4 Rough Sets

Rough Set theory has been applied in such fields as machine learning, data mining, etc., successfully since Professor *Pawlak* developed it in 1982. Reduction of decision table is one of the key problem of rough set theory. The methodology is concerned with the classificatory analysis of imprecise, uncertain or incomplete information or knowledge expressed in terms of data acquired from experience. The primary notions of the theory of rough sets are the approximation space and lower and upper approximations of a set. The approximation space is a classification of the domain of interest into disjoint categories. The membership status with respect to an arbitrary subset of the domain may not always be clearly definable. This fact leads to the definition of a set in terms of lower and upper approximations [9,10,11].

4.1 The Basic Notions

One of the basic fundaments of rough set theory is the *indiscernibility relation* which is generated using information about particular objects of interest. Information about objects is represented in the form of a set of attributes and their associated values for each object. The indiscernibility relation is intended to express the fact that, due to lack of knowledge, we are unable to discern some objects from others simply by employing the available information about thos objects. Any set of all indiscernible (similar) objects is called an *elementary set*, and forms a basic granule (atom) of knowledge about the universe. Any union of some elementary sets in a universe is referred to as a *crisp set.* Otherwise the set is referred to as being a *rough set.* Then, two separate unions of elementary sets can be used to approximate the imprecise set. Vague or imprecise concepts in contrast to precise concepts, cannot be characterized solely in terms of information about their elements since elements are not always discernable from each other. There is an assumption that any vague or imprecise concept is replaced by a pair of precise concepts called the lower and the upper approximation of the vague or imprecise concept.

4.2 Lower/Upper Approximation

The *lower approximation* is a description of the domain objects which are known with certainty to belong to the subset of interest, whereas the *upper approximation* is a description of the objects which possibly belong to the subset. Any subset defined through its lower and upper approximations is called a rough set. It must be emphasized that the concept of rough set should not be confused with the idea of fuzzy set as they are fundamentally different, although in some sense complementary, notions. Rough set approach allows to precisely define the notion of concept approximation. It is based on the *indiscernibility relation* between objects defining a partition of the universe U of objects. The indiscernibility of objects follows from the fact that they are perceived by means of values of available attributes. Hence some objects having the same (or similar) values of attributes are indiscernible.

Let $S = (U, C \cup D)$ be an information system, then with any $B \subseteq C$ there is associated an equivalence relation $IND_S(B)$, called the *B-indiscernibility relation*, its classes are denoted by $[x]_B$.

For $B \subseteq C$ and $X \subseteq U$, we can approximate X using only the information contained in B by constructing the *B-lower*($\underline{BX}$) and *B-upper approximations of* $X(\overline{BX})$, where:

$$\underline{BX} = \{x : [x]_B \subseteq X\}$$

and

$$\overline{BX} = \{x : [x]_B \cap X \neq \emptyset\}.$$

The B-lower approximation of X is the set of all objects which can be certainly classified to X using attributes from B. The difference between the *upper* and the *lower approximation* constitutes the *boundary region* of a vague or imprecise concept. Upper and lower approximations are two of the basic operations in *rough set* theory.

4.3 *Reduct* and *core* of Attributes

In the *rough set* area there is also a very important problem with finding (select) relevant features (attributes), which source is denoted as so called *core* of the information system S.

Reduct is a minimal set of attributes $B \subseteq C$ such that $IND_S(B) = IND_S(C)$, which means that it is a minimal set of attributes from C that preserves the original classification defined by the set C of attributes.

The intersection of all reducts is the so-called *core*. In the example both the *core* and the reduct consist of attributes b and c ($CORE(C) = \{b, c\}$, $RED(C) = \{b, c\}$).

4.4 Rule Induction

Rough set based rule induction methods have been applied to knowledge discovery in databases, whose empirical results obtained show that they are very powerful and that some important knowledge has been extracted from databases. For rule induction, lower/upper approximations and reducts play important roles and the approximations can be extended to variable precision model, using accuracy and coverage for rule induction have never been discussed. We can use the *indiscernibility function* f_S, that form a minimal decision rule for given decision table [1].

For an information system $S = (U, C \cup \{d\})$ with n objects, the *discernibility matrix* of S is a symmetric $n \times n$ matrix with entries c_{ij} defined as:

$$c_{ij} = \{a \in C | a(x_i) \neq a(x_j)\} \text{ for } i, j = 1, 2, \ldots, n$$

where $d(x_i) \neq d(x_j)$). Each entry consists of the set of attributes upon which objects x_i and x_j differ.

A *discernibility function* f_S for an information system S is a boolean function of m boolean variables $a_1^*, \ldots, a_m^*$ (corresponding to the attributes $a_1, \ldots, a_m$) defined by:

$$f_S = \bigwedge \left\{ \bigvee c_{ij}^* | 1 \leq j \leq i \leq n, c_{ij} \neq \emptyset \right\} \tag{9}$$

where $c_{ij}^* = \{a^* | a \in c_{ij}\}$.

For given decision table we formed following set of rules:

– rule nr 1: $\mathit{if}\ a = a_1\ \mathit{and}\ b = b_1 \mathit{then}\ d = T$
– rule nr 2: $\mathit{if}\ b = b_1\ \mathit{and}\ c = c_1 \mathit{then}\ d = T$
– rule nr 3: $\mathit{if}\ b = b_1\ \mathit{and}\ c = c_2 \mathit{then}\ d = T$
– rule nr 4: $\mathit{if}\ c = c_3\ \mathit{then}\ d = T$
– rule nr 5: $\mathit{if}\ c = c_4\ \mathit{then}\ d = N$
– rule nr 6: $\mathit{if}\ b = b_2\ \mathit{and}\ c = c_1 \mathit{then}\ d = N$
– rule nr 7: $\mathit{if}\ c = c_2\ \mathit{then}\ d = T$.

4.5 Rough Set Theory and Decision Systems in Practise

The main specific problems addressed by the theory of rough sets are not only representation of uncertain or imprecise knowledge, or knowledge acquisition from experience, but also the analysis of conflicts, the identification and evaluation of data dependencies and the reduction of the amount of information. A number of practical applications employing this approach have been developed in recent years in areas such as medicine, drug research, process control and other. The recent publication of a monograph on the theory and a handbook on applications facilitate the development of new applications. One of the primary applications of rough sets in artificial intelligence is *knowledge analysis* and *data mining* [12,13,16,17].

From two expert systems implemented at the Silesian University, MEM is the one with the decision table in the form of the knowledge base. It is a diagnosis support system used in child neurology and it is a notable example of a complex multistage diagnosis process. It permits the reduction of attributes, which allows improving the rules acquired by the system. MEM was developed on the basis of real data provided by the Second Clinic of the Department of Paediatrics of the Silesian Academy of Medicine. The system is employed there to support the classification of children having mitochondrial encephalopathies and considerably reduces the number of children directed for further invasive testing in the consecutive stages of the diagnosis process [18,19]. The work contains an example of applying the rough sets theory to application of support decision making. The created system limits maximally the indications for invasive diagnostic methods that finally decide about diagnosis. System has arisen using induction (machine learning from examples) - one of the methods artificial intelligence. Three stages classification has been created. The most important problem was to create an appropriate choice of attributes for the classification process and the generation of a set of rules, a base to make decisions in new cases. Rough set theory provides the appropriate methods which form to solve this problem. A detailed analysis of the medical problem results in creating a three -staged diagnostic process, which allows to classify children into suffering from mitochondrial encephalomyopathy and ones suffering from other diseases. Data on which the decisions were based, like any real data, contained errors. Incomplete information was one of them. It resulted from the fact that some observation or examinations were not possible to be made for all patients. Inconsistency of information was another problem. Inconsistency occured because there were patients who were differently diagnosed at the same values of the parameters analyzed. Additionally developing a supporting decision system in diagnosing was connected with reducing of knowledge, generating decision rules and with a suitable classification of new information.

The first stages of research on decision support systems concentrated on: methods to represent the knowledge in a given system and the methods of the verification and validation of a knowledge base [14].

Recent works, however, deal with the following problems: a huge number of rules in a knowledge base with numerous premises in each rule, a large set of attributes, many of which are dependent, complex inference processes and the problem of the proper interpretation of the decision rules by users. Fortunately, the cluster analysis brings very useful techniques for the smart organisation of the rules, one of which is a hierarchical structure. It is based on the assumption that rules that are similar can be placed in one group. Consequently, in each inference process we can find the most similar group and obtain the forward chaining procedure on this, significantly smaller, group only. The method reduces the time consumption of all processes and explores only the new facts that are actually necessary rather then all facts that can be retrieved from a given knowledge base.

In our opinion, clustering rules for inference processes in decision support systems could prove useful to improve the efficiency of those systems [3,4]. The very important issue for knowledge base modularization is the concept proposed in [26], where the *decision units* conception was presented. Both methods: *cluster analysis* and *decision unit* are subject of our recent researches. We propose such methods to represent knowledge in composited (large, complex) knowledge bases. Using modular representation we can limit the number of rules to process during the inference. Thanks to properties of the cluster and the decision units we can perform different large knowledge bases are an important problem in decision systems. It is well known that the main problem of forward chaining is that it fires a lot of rules, that are unnecessary to fire, because they aren't the inference goal. A lot of fired rules forming a lot of new facts that are difficult to interpret them properly. That is why the optimization of the inference processes in rule based systems is very important in artificial intelligence area. Fortunately there are some methods to solve such problem. For example, we may reorganize the knowledge base from list of not related rules, to groups of similar rules (thanks to *cluster analysis* method) or *decision units*. Thanks to this it is possible to make the inference process (even for really large and composited knowledge bases) very efficient. Simplifying, when we clustering rules, then in inference processes we search only small subset of rules (cluster), that the most similar to given facts or hipothesis [25]. In case of using *decision units* concept, thanks to constructed such units, in *backward chaining* technique we make inference process only on proper decision unit (that with the given conclusion attribute). That is why we propose to change the structure of knowledge base to *cluster* or *decision unit* structure inference algorithm optimizations, depending on user requirements. On this stage of our work we can only present the general conception of modular rule base organization. We can't formally proof that our conception really will cause growth of efficiency. But in our opinion hierarchical organization of rule knowledge base allow us to decrease the number of rules necessary to process during inference, thus we hope that global inference efficiency will grow. On this stage of our reaserch, decision units (with Petri nets extensions) and rules clusters are parallel tools for rule base decomposition rather than a one coherent approach. Therefore we have two methods of rule base decomposition — into the rules clusters if we want to perform forward chaning inference and into the decision units, if we want to do backward chanining inference. The main goal of our future work is to create coherent conception of modularization of large rule bases. This conception shall join two main subgoals: optimalization of forward and backward chaining inference process and practical approach for rule base modelling and verification. In our opinion, two methods of rule base decomposition described in this work, allow as to obtain our goals. It is very important, that exists software tools, dedicated for rules clustering and decision units approach. Practical tests allow us to say, that we need specialized software tools when we work with large, composited rule bases. We expect, that our mixed approach is base for creating such software tools.

Rough sets theory enables solving the problem of a huge number of attributes and dependent attributes removal. The accuracy of classification can be increased by selecting subsets of strong attributes, which is performed by using several classification learners. The processed data are classified by diverse learning schemes and the generation of rules is supervised by domain experts. The implementation of this method in automated decision support software can improve the accuracy and reduce the time consumption as compared to full syntax analysis [20,21,22]. *Pawlak's* theory is also widely used by *Zielosko* and *Piliszczuk* to build clasifiers based on partial reducts and partial decision rules [43,44]. Recently, partial reducts and partial decision rules were studied intesively by *Moshkov* and also *Zielosko* and *Piliszczuk*. Partial reducts and partial decision rules depend on the noise in less degree than exact reducts and rules [42]. Moreover, it is possible to construct more compact classifiers based on partial reducts and rules. The experiments with classifiers presented in [45] show that accuracy of classifiers based on such reducts and rules is often better than the accuracy based on extact reducts and rules.

The very important facts are that in a 1976 Dempster and Shafer have created a mathematical theory of evidence called Dempster-Shafer theory, which is based on belief functions and plausible reasoning [32]. It lets to combine separate pieces of information (evidence) to calculate the probability of an event. Pawlak's rough set theory as an innovative mathematical tool created in 1982 let us to describe the knowledge, including also the uncertain and inexact knowledge [8]. Finally, In 1994 the basic functions of the evidence theory have been defined, based on the notion from the rough set theory [33]. All the dependences between these theories has allowed further research on their practical usage. There are some papers that tried to show the relationships between the rough set theory and the evidence theory which could be used to find the minimal templates for a given decision table were also published [34,35]. Extracting the templates from data is a problem that consists in the finding some set of attributes with a minimal number of attributes, that warrants, among others, the sufficiently small difference between the belief function and the plausibility function. This small difference between these functions allows reducing the number of the attributes (together with the decrease in the vales of the attributes) and made the templates. Moreover MTP (minimal templates problem) gives the recipe witch decision value may be grouped. At the end we get decision rules with the suitable large support.

Of course, in recent years it is possible to witness a rapid growth of interest of application of rought set theory in many other domains such as, for instance, vibration analysis, conflict resolution, intelligent agents, pattern recognition, control theory, signal analysis,process industry, marketing, etc. *Swiniarski* in [36] presented an application of rough sets method to feature selection and reduction as a front end of neural network based texture images recognition. The role of the rough sets is to show its ability to select reduced set of pattern's features.

In other paper, presented by *Nguyen* we can observe the multi-agent system based on rough set theory [37]. The task of creating effective methods of web search result connected with the clustering method, based on rough sets was presented in [41] by *Nguyen. Pawlak's* theory was also used to perform new methodology for data mining in distributed and multiagent systems [38].

Recently, rough set based methods have been proposed for data mining in very large relational data bases [39,40].

4.6 Conclusions

Classification is an important problem in the field of Data Mining. Data acquisition and warehousing capabilities of computer systems are sufficient for wide application of computer aided Knowledge Discovery. Inductive learning is employed in various domains such as medical data analysis or customer activity monitoring. Due to various factors that data suffer from impreciseness and incompleteness. There are many classification approaches like "nearest neighbours", "naive Bayes", "decision tree", "decision rule set", "neural networks" and many others. Unfortunately, there are opinions that rough set based methods can be used for small data set only. The main approach is related to their lack of scalability (more precisely: there is a lack of proof showing that they can be scalable). The biggest troubles stick in the rule induction step. As we know, the potential number of all rules is exponential. All heuristics for rule induction algorithms have at least $O(n^2)$ time complexity, where n is the number of objects in the data set and require multiple data scanning. Rough Sets Theory has been applied to build classifiers by exploring symbolic relations in data. Indiscernibility relations combined with the cloncept notion, and the application of set operations, lead to knowledge discovery in an elegant and intuitive way. Knowledge discovered from data talbes is often presented in terms of "if...then..." decision rules. With each rule a confidence measure is associated. Rough sets provide symbolic representation of data and the representation of knowledge in terms of aatributes, information tables, semantic decision rules, rough measures of inclusion and closeness of information granules, and so on. Rough set methods make possible to reduce the size of a dataset by removing some of the attributes while preserwing the partitioning of the universe of an information system into equivalence classes.

5 Summary

Information systems and decision support systems are strongly related. The paper shows that we can treat a decision system as an information system of some objects, for which we have the information about their classification. When the information is not complete, or the system has some uncertain data - we can use rough set theory to separate the uncertain part from that, what we are sure about. By defining the *reduct* for a decision table, we can optimize the system and then, using the methods for minimal rules generation, we can easily classify new objects. We see, therefore, that *Prof. Pawlak's* contribution to the domain of information and decision support systems is invaluable [24].

References

1. Bazan, J.: Metody wnioskowań aproksymacyjnych dla syntezy algorytmów decyzyjnych, praca doktorska, Wydzia l Informatyki, Matematyki i Mechaniki, Uniwersytet Warszawski, Warszawa (1998)
2. Grzelak, K., Kochańska, J.: System wyszukiwania informacji metodą składowych atomowych MSAWYSZ, ICS PAS Reports No 511, Warsaw (1983)
3. Nowak, A., Wakulicz-Deja, A.: Effectiveness comparison of classification rules based on k-means Clustering and Salton's Method. In: Advances in Soft Computing, pp. 333–338. Springer, Heidelberg (2004)
4. Nowak, A., Wakulicz-Deja, A.: The concept of the hierarchical clustering algorithms for rules based systems. In: Advances in Soft Computing, pp. 565–570. Springer, Heidelberg (2005)
5. Pawlak, Z.: Mathematical foundation of information retrieval. CC PAS Reports No 101, Warsaw (1973)
6. Pawlak, Z., Marek, W.: Information Storage and retrieval system-mathematical foundations. CC PAS Reports No. 149, Warsaw (1974)
7. Pawlak, Z.: Information systems theoretical foundation. Information Systems 6(3) (1981)
8. Pawlak, Z.: Rough Sets: Theoretical aspects of reasoning about data. Kluwer Academic Publishers, Boston (1991)
9. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Information Sciences 177, 3–27 (2007)
10. Pawlak, Z., Skowron, A.: Rough sets: some extensions. Information Sciences 177, 28–40 (2007)
11. Pawlak, Z., Skowron, A.: Rough sets and Boolean reasoning. Information Sciences 177, 41–73 (2007)

12. Roddick, J.F., Hornsby, K., Spiliopoulou, M.: YABTSSTDMR - Yet Another Bibliography of Temporal, Spatial and Spatio-Temporal Data Mining Research. In: Unnikrishnan, K.P., Uthurusamy, R. (eds.) Proc. SIGKDD Temporal Data Mining Workshop, San Francisco, CA, pp. 167–175. ACM, New York (2001)
13. Roddick, J.F., Egenhofer, M.J., Hoel, E., Papadias, D., Salzberg, B.: Spatial, Temporal and Spatio-Temporal Databases - Hot Issues and Directions for Ph.D Research. SIGMOD Record 33(2), 126–131 (2004)
14. Simiński, R., Wakulicz-Deja, A.: Circularity in Rule Knowledge Bases - Detection using Decision Unit Approach. In: Advances in Soft Computing, pp. 273–280. Springer, Heidelberg (2004)
15. Skowron, A.: From the Rough Set Theory to the Evidence Theory. In: Yager, R.R., Fedrizzi, M., Kacprzyk, J. (eds.) Advances in the Dempster-Shafer Theory of Evidence, pp. 193–236. Wiley, New York (1994)
16. Skowron, A., Bazan, J., Stepaniuk, J.: Modelling Complex Patterns by Information Systems. Fundamenta Informaticae 67(1-3), 203–217 (2005)
17. Bazan, J., Peters, J., Skowron, A., Synak, P.: Spatio-temporal approximate reasoning over complex objects. Fundamenta Informaticae 67, 249–269 (2005)
18. Wakulicz-Deja, A.: Podstawy systemów ekspertowych. Zagadnienia implementacji. Studia Informatica 26(3(64)) (2005)
19. Wakulicz-Deja, A., Paszek, P.: Optimalization on Decision Problems on Medical Knowledge Bases. In: 5th European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany (1997)
20. Wakulicz-Deja, A., Ilczuk, G.: Attribute Selection and Rule Generation Techniques for Medical Diagnosis Systems. In: Ślęzak, D., Yao, J., Peters, J.F., Ziarko, W.P., Hu, X. (eds.) RSFDGrC 2005. LNCS, vol. 3642, pp. 352–361. Springer, Heidelberg (2005)
21. Wakulicz-Deja, A., Ilczuk, G., Kargul, W., Mynarski, R., Drzewiecka, A., Pilat, E.: Artificial intelligence in echocardiography - from data to conclusions. Eur. J. Echocardiography Supplement 7(supl.1) (2006)
22. Wakulicz-Deja, A., Paszek, P.: Applying rough set theory to multi stage medical diagnosing. Fundamenta Informaticae XX, 1–22 (2003)
23. Lum, V.Y.: Multi-Attribute Retrieval with Combined Indexes. Communications of the ACM 13(11) (1970)
24. Wakulicz-Deja, A., Nowak, A.: From an information system to a decision support system. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 454–464. Springer, Heidelberg (2007)
25. Nowak, A., Wakulicz-Deja, A.: The inference processes on clustered rules. In: Advances in Soft Computing, vol. 5, pp. 403–411. Springer, Heidelberg (2006)
26. Nowak, A., Simiński, R., Wakulicz-Deja, A.: Towards modular representation of knowledge base. In: Advances in Soft Computing, vol. 5, pp. 421–428. Springer, Heidelberg (2006)
27. Effelsberg, W., Harder, T., Reuter, A.: An experiment in learning DBTG data-base administration. Information Systems 5, 137–147 (1980)
28. Michalski, R.S., Chilausky, R.L.: Knowledge acquisition by encoding expert rules versus computer indution from examples: a case study involving soybean pathology. International Journal Man. Machine Studies 12, 63–87 (1977)
29. Slamecka, V., Comp, H.N., Bodre, A.: MARIS - A knowledge system for internal medicine. Information Processing and Management 5, 273–276 (1977)
30. Masui, S., Shioya, M., Salaniski, T., Tayama, Y., Iungawa, T.: Fujite: Evaluation of a diffusion model applicable to environmental assessment for air polution abatement, System Development Lab. Hitachi Ltd., Tokyo, Japan (1980)

31. Cash, J., Whinston, A.: Security for GPLAN system. Information Systems 2(2) (1976)
32. Shafer, G.: A mathematical theory of evidence. Princeton University Press, Princeton (1976)
33. Skowron, A., Grzymala-Busse, J.: From the Rough Set Theory to the Evidence Theory. In: Yager, R.R., Fedrizzi, M., Kacprzyk, J. (eds.) Advances in the Dempster-Shafer Theory of Evidence, pp. 193–236. Wiley, New York (1994)
34. Marszal-Paszek, B., Paszek, P.: Minimal Templates Problem, Intelligent Information Processing and Web Mining. In: Advances in Soft Computing, vol. 35, pp. 397–402. Springer, Heidelberg (2006)
35. Marszal-Paszek, B., Paszek, P.: Extracting Minimal Templates in a Decision Table, Monitoring, Security, and Rescue Techniques in Multiagent Systems. In: Advances in Soft Computing, vol. 2005, pp. 339–344. Springer, Heidelberg (2005)
36. Swiniarski, R., Hargis, L.: Rough Sets as a Front End of Neural Networks Texture Classifiers. Neurocomputing 36(1-4), 85–102 (2001)
37. Nguyen, H.S., Nguyen, S.H., Skowron, A.: Decomposition of Task Specification. In: Raś, Z.W., Skowron, A. (eds.) ISMIS 1999. LNCS, vol. 1609, p. 310. Springer, Heidelberg (1999)
38. Polkowski, L., Skowron, A. (eds.): Rough Sets in Knowledge Discovery 2: Applications, Case Studies and Software Systems. Physica Verlag, Heidelberg (1998)
39. Stepaniuk, J.: Relational Data and Rough Sets. Fundamenta Informaticae 79(3-4), 525–539 (2007)
40. Stepaniuk, J.: Approximation Spaces in Multi Relational Knowledge Discovery. Rough Sets 6, 351–365 (2007)
41. Ngo, C.L., Nguyen, H.S.: A Method of Web Search Result Clustering Based on Rough Sets, Web Intelligence, pp. 673–679 (2005)
42. Moshkov, M., Piliszczuk, M., Zielosko, B.: On construction of partial reducts and irreducible partial decision rules. Fundamenta Informaticae 75(1-4), 357–374 (2007)
43. Piliszczuk, M.: On greedy algorithm for partial reduct construction. In: Proceedings of concurency, Specification and Programming Workshop, Ruciane Nida, Poland, pp. 400–411 (2005)
44. Zielosko, B.: On partial decision rules. In: Proceedings of concurency, Specification and Programming Workshop, Ruciane Nida, Poland, pp. 598–609 (2005)
45. Zielosko, B., Kocjan, A., Piliszczuk, M.: Classifiers Based on Partial Reducts and Partial Decision Rules. In: Intelligent Information Systems XVI. Proceedings of the International IIS 2008 Conference held in Zakopane. Challenging Problems of Science. Computer Science, pp. 431–438. Academic Publishing House EXIT, Warsaw (2008)
46. Orlowska E.: Dynamic information systems, IPI PAN papers Nr. 434, Warsaw, Poland (1981)
47. Sarnadas, A.: Temporal aspects of logical procedure definition. Information Systems (3) (1980)
48. Wakulicz-Deja, A.: Classification of time - varying information systems. Information Systems (3), Warsaw, Poland (1984)
49. Wakulicz-Deja, A.: Podstawy systemów wyszukiwania informacji. Analiza Metod, Problemy Współczesnej Nauki. Teoria i Zastosowania. Informatyka, Akademicka Oficyna Wydawnicza PLJ, Warsaw, Poland (1995)

Debellor: A Data Mining Platform with Stream Architecture

Marcin Wojnarski

Warsaw University, Faculty of Mathematics, Informatics and Mechanics
ul. Banacha 2, 02-097 Warszawa, Poland
mwojnars@ns.onet.pl

Abstract. This paper introduces Debellor (www.debellor.org) – an open source extensible data mining platform with stream-based architecture, where all data transfers between elementary algorithms take the form of a stream of samples. Data streaming enables implementation of *scalable* algorithms, which can efficiently process large volumes of data, exceeding available memory. This is very important for data mining research and applications, since the most challenging data mining tasks involve voluminous data, either produced by a data source or generated at some intermediate stage of a complex data processing network.

Advantages of data streaming are illustrated by experiments with clustering time series. The experimental results show that even for moderate-size data sets streaming is indispensable for successful execution of algorithms, otherwise the algorithms run hundreds times slower or just crash due to memory shortage.

Stream architecture is particularly useful in such application domains as time series analysis, image recognition or mining data streams. It is also the only efficient architecture for implementation of online algorithms.

The algorithms currently available on Debellor platform include all classifiers from Rseslib and Weka libraries and all filters from Weka.

Keywords: Pipeline, Online Algorithms, Software Environment, Library.

1 Introduction

In the fields of data mining and machine learning, there is frequently a need to process large volumes of data, too big to fit in memory. This is particularly the case in some application domains, like computer vision or mining data streams [1,2], where input data are usually voluminous. But even in other domains, where input data are small, they can abruptly expand at an intermediate stage of processing, e.g. due to extraction of windows from a time series or an image [3,4]. Most of ordinary algorithms are not suitable for such tasks, because they try to keep all data in memory. Instead, special algorithms are necessary, which make efficient use of memory. Such algorithms will be called *scalable*.

Another feature of data mining algorithms – besides scalability – which is very desired nowadays is *interoperability*, i.e. a capability of the algorithm to be easily connected with other algorithms. This property is more and more important, as basically all newly created data mining systems – whether experimental or end-user solutions – incorporate much more than just one algorithm.

It would be very worthful if algorithms were both scalable and interoperable. Unfortunately, combining these two features is very difficult. Interoperability requires that every algorithm is implemented as a separate module, with clearly defined input and output. Obviously, data mining algorithm must take data as its input, so the data must be fully *materialized* – generated and stored in a data structure – just to invoke the algorithm, no matter what it actually does. And materialization automatically precludes scalability of the algorithm.

In order to provide scalability and interoperability at the same time, algorithms must be implemented in a special software architecture, which do not enforce data materialization. Debellor[1] – the data mining platform introduced in this paper – defines such an architecture, based on the concept of *data streaming*. In Debellor, data are passed between interconnected algorithms sample-by-sample, as a stream of samples, so they can be processed on the fly, without full materialization. The idea of data streaming is inspired by architectures of database management systems, which enable fast query execution on very large data tables.

It should be noted that Debellor is not a library, like e.g., Rseslib[2] [5,6,7] or Weka[3] [8], but a data mining *platform*. Although its distribution contains implementations of a number of algorithms, the primary goal of Debellor is to provide not algorithms themselves, but a common architecture, in which various types of data processing algorithms may be implemented and combined, even if they are created by independent researchers. Debellor can handle a wide range of algorithm types: classifiers, clusterers, data filters, generators etc. Moreover, extendability of data types is provided, so it will be possible to process not only ordinary feature vectors, but also images, text, DNA microarray data etc.

It is worth mentioning that Debellor's modular and stream-oriented architecture will enable easy parallelization of composite data mining algorithms. This aspect will be investigated elsewhere.

Debellor is written in Java and distributed under GNU General Public License. Its current version, Debellor 0.5, is available at www.debellor.org. The algorithms currently available include all classifiers from Rseslib and Weka libraries, all filters from Weka and a reader of ARFF files. There are also several algorithms implemented by Debellor itself, like Train&Test evaluation procedure. The algorithms from Rseslib and Weka, except the ARFF reader, are not scalable – this is enforced by architectures of both libraries.

[1] The name originates from Latin *debello* (to conquer) and *debellator* (conqueror).
[2] http://rsproject.mimuw.edu.pl/
[3] http://www.cs.waikato.ac.nz/ml/weka/

2 Related Work

There is large amount of software that can be used to facilitate implementation of new data mining algorithms. A common choice is to use an environment for numerical calculations: R[4] [9], Matlab[5], Octave[6] [10,11] or Scilab[7] and implement the algorithm in a scripting language defined by the environment. Many data mining and machine learning algorithms are available for each of these environments, usually in a form of external packages, so the environments can be seen as common platforms for different data mining algorithms. However, they do not define common architecture for algorithms, so they do not automatically provide interoperability. Moreover, the scripting languages of these environments have low efficiency, no static typing and only weak support for object-oriented programming, so they are suitable for fast prototyping and running small experiments, but not for implementation of scalable and interoperable algorithms.

Another possible choice is to take a data mining library written in a general-purpose programming language (usually Java) – examples of such libraries are Weka[8] [8], Rseslib[9] [5,6,7], RapidMiner[10] [12] – and try to fit the new algorithm into the architecture of the library. However, these libraries preclude scalability of algorithms, because the whole training data must be materialized in memory before they can be passed to an algorithm.

The concept of data streaming, called also pipelining, has been used in database management systems [13,14,15,16] for efficient query execution. The elementary units capable of processing streams are called *iterators* in [13,14].

The issue of scalability is related to the concept of online algorithms. In machine learning literature [17,18], the term *online* has been used to denote training algorithms which perform updates of the underlying decision model after every single presentation of a sample. The algorithms which update the model only when the whole training set has been presented are called *batch*.

Usually online algorithms can be more memory-efficient than their batch counterparts, because they do not have to store samples for later use. They are also more flexible, e.g., they can be used in incremental learning or allow for the training process to be stopped anytime during scan of the data. This is why extensive research has been done to devise online variants of existing batch algorithms [19,20,21,22,23]. Certainly, online algorithms are the best candidates for implementation in stream architecture. Note, however, that many batch algorithms also do not have to keep all samples in memory and thus can benefit from data streaming. In many cases it is enough to keep only some statistics calculated during scan of the data set, used afterwards to make the final update of the model. For example, standard k-means [17,24,25] algorithm performs batch

[4] http://www.r-project.org
[5] http://www.mathworks.com
[6] http://www.octave.org
[7] http://www.scilab.org
[8] http://www.cs.waikato.ac.nz/ml/weka
[9] http://rsproject.mimuw.edu.pl
[10] http://rapid-i.com

updates of the model, but despite this it can be scalable if implemented in stream architecture, as will be shown in Sect. 5.8.

3 Motivation

3.1 Scalability

Scalable algorithms are indispensable in most of data mining tasks – every time when data become larger than available memory. Even if initially memory seems capacious enough to hold the data, it may appear during experiments that data are larger and memory smaller than expected. There are many reasons for this:

1. Not the whole physical memory is available to the data mining algorithm at a given time. Some part is used by operating system and other applications.
2. The experiment may incorporate many algorithms run in parallel. In such case, available memory must be partitioned between all of them. In the future, parallelization will become more and more common due to parallelization of hardware architectures, e.g., expressed by increasing number of cores in processors.
3. In a complex experiment, composed of many elementary algorithms, every intermediate algorithm will generate another set of data. Total amount of data will be much larger than the amount of source data alone.
4. For architectural reasons data must be stored in memory in some general data structures, which take more memory than would be necessary in a given experiment. For example, data may be composed of binary attributes and each value could be stored on a single bit, but in fact each value takes 8 bytes or more, because every attribute – whether it is numeric or binary – is stored in the same way. Internal data representation used by a given platform is always a compromise between generality and efficient memory usage.
5. Data generated at intermediate processing stages may be many times larger than source data. For example:
   - Input data may require decompression, e.g. JPEG images must be converted to raw bitmaps to undergo processing. This may increase data size even by a factor of 100.
   - In image recognition, a single input image may be used to generate thousands of subwindows that would undergo further processing [4,26]. An input image of 1MB size may easily generate windows of 1GB size or more. Similar situation occurs in speech recognition or time series analysis, where the sliding-window technique is used.
   - Synthetic attributes may be generated, e.g. by taking all multiplications of pairs of original attributes, which leads to quadratic increase in the number of attributes.
   - Synthetic samples may be generated, in order to increase the size of training set and improve learning of a decision system. For example, this method is used in [27], which studies the problem of Optical Character Recognition. Training images of hand-written characters are randomly

distorted by planar affine transformations and added to the training set. Every image undergoes 9 random distortions, which leads to 10-fold increase in the training set size (from 60 to 600 thousand images).

6. In some applications, like mining data streams [1], input data are potentially infinite, so scalability obviously becomes an issue.
7. Even if the volume of data is small at the stage of experiments, it may become much bigger when the algorithm is deployed in a final product and must process real-world instead of experimental data.

The above arguments show clearly that memory is indeed a critical issue for data mining algorithms. Every moderately complex experiment will show one or more of the characteristics listed above. This is why we need scalable algorithms and – for this purpose – an architecture that will enable algorithms to process data on the fly, without full materialization of a data set.

3.2 Interoperability

Nowadays, it is impossible to solve a data mining task or conduct an experiment using only one algorithm. For example, even if you want to experiment with a single algorithm, like a new classification method, you at least have to access data on disk, so you need an algorithm that reads a given file format (e.g. ARFF[11]). Also, you would like to evaluate your classifier, so you need an algorithm which implements an evaluation scheme, like cross-validation or bootstrap. And in most cases you will also need several algorithms for data preprocessing like normalization, feature selection, imputation of missing values etc. – note that preprocessing is an essential step in knowledge discovery [28,29] and usually several different preprocessing methods must be applied before data can be passed to a decision system.

To build a data mining system, there must be a way to connect all these different algorithms together. Thus, they must possess the property of interoperability. Without this property, even the most efficient algorithm is practically useless.

Further on, the graph of data flow between elementary algorithms in a data mining system will be called a *Data Processing Network* (DPN). In general, we will assume that DPN is a directed acyclic graph, so there are no loops of data flow. Moreover, in the current version of Debellor, DPN can only have a form of a single chain, without branches. An example of DPN is shown in Figure 1.

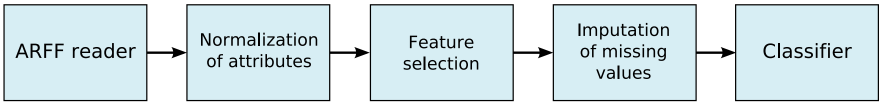


Fig. 1. Example of a Data Processing Network (DPN), composed of five elementary algorithms (boxes). Arrows depict data flow between the algorithms.

[11] http://www.cs.waikato.ac.nz/ml/weka/arff.html

4 Data Streaming

To provide interoperability, data mining algorithms must be implemented in a common software *architecture*, which specifies:

– a method for connecting algorithms,
– a model of data transfer,
– common data representation.

Architectures of existing data mining systems utilize the *batch* model of data transfer. In this model, algorithms must take the whole data set as an argument for execution. To run composite experiment, represented by a DPN with a number of algorithms, an additional *supervisor* module is needed, responsible for invoking consecutive algorithms and passing data sets between them. Figure 3 presents a UML *sequence diagram* [30] with an example of batch processing in a DPN composed of three algorithms. DPN itself is presented in Fig. 2.

Batch data transfer enforces data materialization, which precludes scalability of algorithms and DPN as a whole. For example, in Weka, every classifier must be implemented as a subclass of `Classifier` class (in `weka.classifiers` package). Its training algorithm must be implemented in the method:

```
buildClassifier(Instances) :  void
```

The argument of type `Instances` is an array of training samples. This argument must be created before calling `buildClassifier`, so the data must be fully materialized in memory just to invoke training algorithm, no matter what the algorithm actually does.

Similar situation takes place for clustering methods, which must inherit from `weka.clusterers.Clusterer` class and overload the method:

```
buildClusterer(Instances) :  void
```

Rseslib and RapidMiner also enforce data materialization before a training algorithm can be invoked. In Rseslib, classifiers must be trained in the class constructor, which takes an argument of type `DoubleDataTable`. In RapidMiner, training of any decision system takes place in the method `apply(IOContainer)` of the class `com.rapidminer.operator.Operator`. Both Rseslib's `DoubleDataTable` and RapidMiner's `IOContainer` represent materialized input data.

If a large data set must be materialized, execution of the experiment is practically impossible. If data fit in virtual memory [31], but exceed available physical memory, operating system temporarily swaps [31] part of the data (stores it in

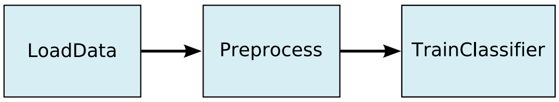


Fig. 2. DPN used as an example for analysis of data transfer models

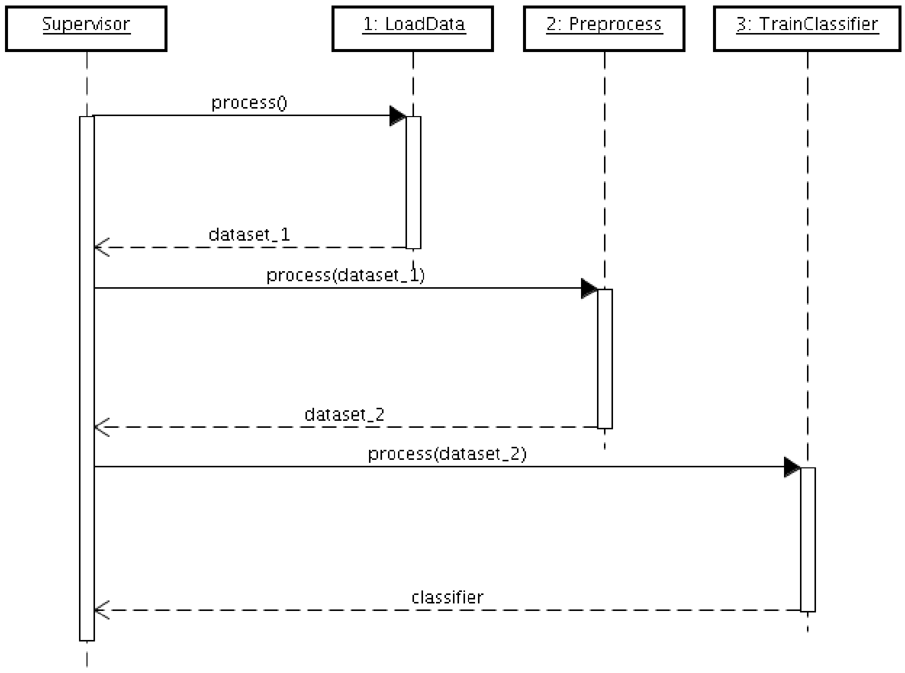


Fig. 3. UML diagram of *batch* data transfer in a DPN composed of three algorithms: LoadData, Preprocess and TrainClassifier, controlled by the Supervisor module. Supervisor invokes the algorithms (methods `run`) and pass data between them. All samples of a given data set are generated and transferred together, so available memory must be large enough to hold all data. Vertical lines denote life of modules, with time passing down the lines. Horizontal lines represent messages (method calls and/or data transfers) between the modules. Vertical boxes depict execution of the module's code.

the swap file on disk), which makes the execution tens or hundreds times slower, as access to disk is orders of magnitude slower than to memory.

If the data set is so large that it even exceeds available virtual memory, execution of the experiment is terminated with an out-of-memory error. This problem could be avoided if the class that represents a data set (e.g., `Instances` in Weka) implemented internally the buffering of data on disk. Then, however, the same performance degradation would occur as in the case of system swapping, because swapping and buffering on disk are actually the same things, only implemented at different levels: of operating system or data mining environment.

The only way to avoid severe performance degradation when processing large data is to generate data iteratively, sample-by-sample, and instantly process created samples, as presented in Fig. 4. In this way, data may be generated and consumed on the fly, without materialization of the whole set. This model of data transfer will be called *iterative*.

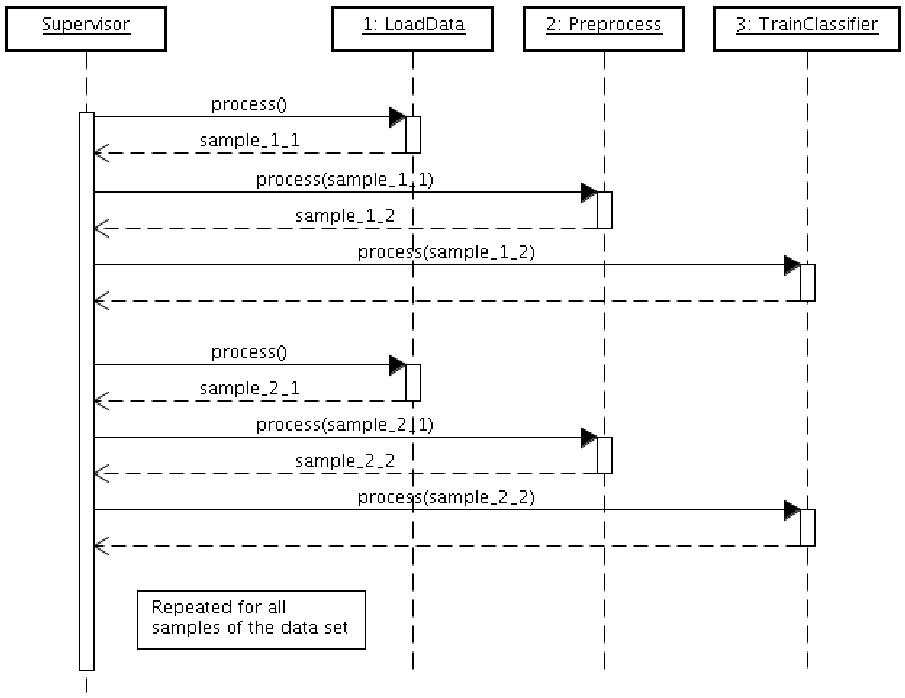


Fig. 4. UML diagram of *iterative* data transfer. The supervisor invokes the algorithms separately for each sample of the data set (sample_x_y denotes sample no. x generated by algorithm no. y). In this way, memory requirements are very low (memory complexity is constant), but supervisor's control over data flow becomes very difficult.

Iterative data transfer solves the problem of high memory consumption, because memory requirements imposed by the architecture are constant – only a fixed number of samples must be kept in memory in a given moment, no matter how large the full data set is. However, another problem arises: the supervisor becomes responsible for controlling the flow of samples and the order of execution of algorithms. This control may be very complex, because each elementary algorithm may have different input-output characteristics. The number of possible variants is practically infinite, for example:

1. Preprocessing algorithm may filter out some samples, in which case more than one input sample may be needed to produce one output sample.
2. Preprocessing algorithm may produce a number of output samples from a single input sample, e.g. when extracting windows from an image or time series.
3. Training algorithm of a decision system usually have to scan data many times, not only once.

4. Generation of output samples may be delayed relatively to the flow of input samples, e.g. an algorithm may require that 10 input samples are given before it starts producing output samples.
5. Input data to an algorithm may be infinite, e.g. when they are generated synthetically. In such case, the control mechanism must stop data generation in appropriate moment.
6. Some algorithms may have more than one input or output, e.g. an algorithm for merging data from several different sources (many inputs) or an algorithm for splitting data into training and test parts (many outputs). In such case, the control of data flow through all the inputs and outputs becomes even more complex, because there are additional dependencies between many inputs/outputs of the same algorithm.

Note that the diagram in Fig. 4 depicts a simplified case when DPN is a single chain of three algorithms, without branches; preprocessing generates exactly one output sample for every input sample; and training algorithm scans data only once.

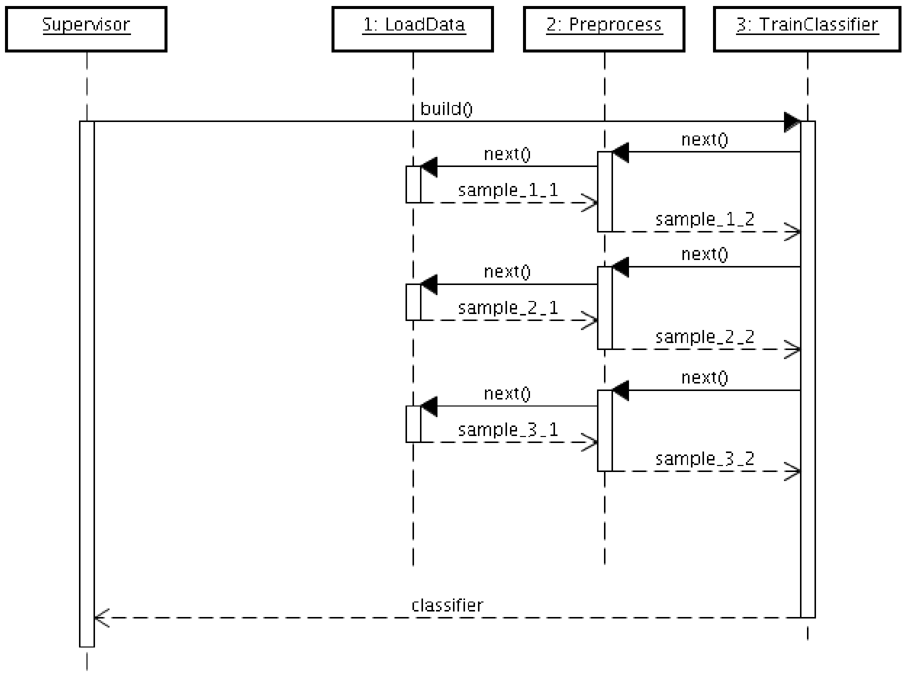


Fig. 5. UML diagram of control and data flow in the *stream* model of data transfer. The supervisor invokes only method build() of the last component (TrainClassifier). This triggers a cascade of messages (calls to methods next()) and transfers of samples, as needed to fulfill the initial build() request.

The way how data flow should be controlled depends on what algorithms are used in a given DPN. For this reason, the algorithms themselves – not the supervisor – should be responsible for controlling data flow. To this end, each algorithm must be implemented as a *component* which can communicate with other components without external control of the supervisor. Supervisor's responsibility must be limited only to linking components together (building DPN) and invoking the last algorithm in DPN, which is the final receiver of all samples. Communication should take the form of a stream of samples: (i) sample is the unit of data transfer; (ii) samples are transferred sequentially, in a fixed order decided by the sender. This model of data transfer will be called a *stream* model. An example of control and data flow in this model is presented in Fig. 5.

Component architecture and data streaming are the features of Debellor which enable scalability of algorithms implemented on this platform.

5 Debellor Data Mining Platform

5.1 Data Streams

Debellor's components are called *cells*. Every cell is a Java class inheriting from the base class `Cell` (package `org.debellor.core`). Cells may implement all kinds of data processing algorithms, for example:

1. Decision algorithms: classification, regression, clustering, density estimation etc.
2. Transformations of samples and attributes.
3. Removal or insertion of samples and attributes.
4. Loading data from file, database etc.
5. Generation of synthetic data.
6. Buffering and reordering of samples.
7. Evaluation schemes: train&test, cross-validation, leave-one-out etc.
8. Collecting statistics.
9. Data visualization.

Cells may be connected into DPN by calling the `setSource(Cell)` method on the receiving cell, for example:

```
Cell cell1 = ..., cell2 = ..., cell3 = ...;
cell2.setSource(cell1);
cell3.setSource(cell2);
```

The first cell will usually represent a file reader or a generator of synthetic data. Intermediate cells may apply different kinds of data transformations, while the last cell will usually implement a decision system or an evaluation procedure.

DPN can be used to process data by calling methods `open()`, `next()` and `close()` on the last cell of DPN, for example:

```
        cell3.open();
        sample1 = cell3.next();
        sample2 = cell3.next();
        sample3 = cell3.next();
        ...
        cell3.close();
```

The above calls open communication session with `cell3`, retrieve some number of processed samples and close the session. In order to realize each request, `cell3` may communicate with its source cell, `cell2`, by invoking the same methods (`open`, `next`, `close`) on `cell2`. And `cell2` may in turn communicate with `cell1`. In this way it is possible to generate output samples on the fly. The stream of samples may flow through consecutive cells of DPN without buffering, so input data may have unlimited volume.

Note that the user of DPN does not have to control sample flow by himself. To obtain the next sample of processed data it is enough to call `cell3.next()`, which will invoke – if needed – a cascade of calls to preceding cells.

Moreover, different cells may control the flow of samples differently. For example, cells that implement classification algorithms will take one input sample in order to generate one output sample. Filtering cells will take a couple of input samples in order to generate one output sample that matches the filtering rule. The image subwindow generator will produce many output samples out of a single input sample. We can see that the cell's interface is very flexible. It enables implementation of various types of algorithms in the same framework and allows to easily combine the algorithms into a complex DPN.

5.2 Buildable Cells

Some cells may be *buildable*, in which case their *content* must be built before the cell can be used. Building procedure is invoked by calling method

```
        build() :  void
```

on the cell object. This method is declared in the base class `Cell`.

Building a cell may mean different things for different types of cells. For example:

– training a decision system of some kind (classifier, clusterer, . . .),
– running an evaluation scheme (train&test, cross-validation, . . .),
– reading all data from input stream and buffering in memory.

Note that all these different types of algorithms are encapsulated under the same interface (method `build()`). This increases simplicity and modularity of the platform.

Usually, the cell reads input data during building, so it must be properly connected to a source cell before `build()` is invoked. Afterwards, the cell may be reconnected and used to process another stream of data.

Some buildable cells may also implement `erase()` method, which clears the content of the cell. After erasure, the cell may be built once again.

5.3 State of the Cell

Every cell object has a *state* variable attached, which indicates what cell operations are allowed in a given moment. There are three possible states: EMPTY, CLOSED and OPEN. Transitions between them are presented in Fig. 6. Each transition is invoked by call to an appropriate method: `build()`, `erase()`, `open()` or `close()`.

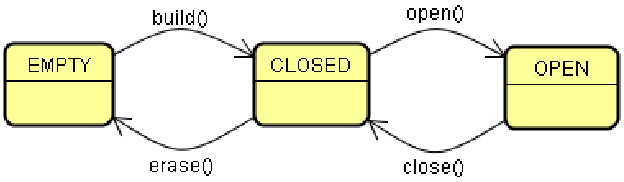


Fig. 6. Diagram of cell states and allowed transitions

Only a part of cell methods may be called in a given state. For example, `next()` can be called only in OPEN state, while `setSource()` is allowed only in EMPTY or CLOSED state. It is guaranteed by the base class implementation that disallowed calls immediately end with an exception thrown. Thanks to this automatic state control, connecting different cells together and building composite algorithms becomes easier and safer, because many possible mistakes or bugs related to inter-cell communication are detected early. Otherwise, they could exist unnoticed, generating incorrect results during data processing. Moreover, it is easier to implement new cells, because the authors do not have to check correctness of method calls by themselves.

5.4 Parametrization

Most of cells require a number of parameters to be set before the cell can start working. Certainly, every type of a cell requires different parameters, but for the sake of interoperability and simplicity of usage, there should be a common interface for passing parameters, no matter what number and types of parameters are expected by a given cell. Debellor defines such an interface.

Parameters for a given cell are stored in an object of class `Parameters` (package `org.debellor.core`), which keeps a dictionary of parameter names and associated `String` values (in the future we plan to extend permitted value types, note however that all simple types can be easily converted to `String`). Thanks to the use of a dictionary, the names do not have to be hard-coded as fields of cell objects, hence parameters can be added dynamically, according to requirements of a given cell.

The object of class `Parameters` can be passed to the cell by calling `Cell`'s method:

```
setParameters(Parameters) :  void
```

It is also possible (and usually more convenient) to pass single parameter values directly to the cell, without an intermediate `Parameters` object, by calling:

```
set(String name, String value) :  void
```

This method call delegates to analogous method of `Cell`'s internal `Parameters` object.

5.5 Data Representation

The basic unit of data transfer between cells is *sample*. Samples are represented by objects of class `Sample`. Every sample contains two fields, `data` and `label`, which hold input data and associated decision label, respectively. Any of the fields can be `null`, if corresponding information is missing or simply not necessary at the given point of data processing.

Cells are free to use whichever part of input data they want. For example, `build()` method of a classifier (i.e. training algorithm) would use both `data` and `label`, interpreting `label` as a target classification of `data`, given by a supervisor. During operation phase, the classifier would ignore input `label`, if present. Instead, it would classify `data` and assign the generated label to the `label` field of the output sample.

Data and labels are represented in an abstract way. Both `data` and `label` fields reference objects of type `Data` (package `org.debellor.core`). `Data` is a base class for classes that represent data items, like single features or vectors of features. When the cell wants to use information stored in `data` or `label`, it must downcast the object to a specific subclass, as expected by the cell. Thanks to this abstract method of data representation, new data types can be added easily, by creating a new subclass of `Data`. Authors of new cells are not limited to a single data type, hard-coded into the platform, as for example in Weka.

Data objects may be nested. For example, objects of class `DataVector` (in `org.debellor.core.data`) hold arrays of other data objects, like simple features (classes `NumericFeature` and `SymbolicFeature`) or other `DataVectors`.

5.6 Immutability of Data

A very important concept related to data representation is *immutability*. Objects which store data – instances of `Sample` class or `Data` subclasses – are *immutable*, i.e. they cannot be modified after creation. Thanks to this property, data objects can be safely shared by cells, without risk of accidental modification in one cell that would affect operations of another cell.

Immutability of data objects yields many benefits:

1. Safety – cells written by different people may work together in a complex DPN without interference.
2. Simplicity – the author of a new cell does not have to care about correctness of access to data objects.

3. Efficiency – data objects do not have to be copied when transferred to another cell. Without immutability, copying would be necessary to provide a basic level of safety. Also, a number of samples may keep references to the same data object.
4. Parallelization – if DPN is executed concurrently, no synchronization is needed when accessing shared data objects. This simplifies parallelization and makes it more efficient.

5.7 Metadata

Many cells have to know some basic characteristics ("type") of input samples before processing of the data starts. For example, training algorithm of a neural network has to know the number of input features, to be able to allocate arrays of weights of appropriate size. To provide such information, method `open()` returns an object of class `MetaSample` (static inner class of `Sample`), which describes common properties of all samples generated by the stream being open. Similarly to `Sample`, `MetaSample` has separate fields describing input data and labels, both of type `MetaData` (static inner class of `Data`).

Metadata have analogous structure and properties as the data being described. The hierarchy of metadata classes, rooted at `MetaData`, mirrors the hierarchy of data classes, rooted at `Data`. The nesting of `MetaData` and `Data` objects is also similar, e.g. if the stream generates `DataVector`s of 10 `SymbolicFeature`s, corresponding `MetaData` object will be an instance of `MetaDataVector`, containing an array of 10 `MetaSymbolicFeature`s describing every feature.

Similarly to `Data`, `MetaData` objects are immutable, so they can be safely shared by cells.

5.8 Example

To illustrate the usage of Debellor, we will show how to implement standard k-means algorithm in stream architecture and how to employ it to data processing in a several-cell DPN.

K-means [17,24,25] is a popular clustering algorithm. Given n input samples – numeric vectors of fixed length, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ – it tries to find cluster centers $\mathbf{c}_1, \ldots, \mathbf{c}_k$ which minimize the sum of squared distances of samples to their closest center:

$$\mathcal{E}(\mathbf{c}_1, \ldots, \mathbf{c}_k) = \sum_{i=1}^{n} \min_{j=1,\ldots,k} \|\mathbf{x}_i - \mathbf{c}_j\|^2 . \tag{1}$$

This is done through iterative process with two steps repeated alternately in a loop: (i) assignment of each sample to the nearest cluster and (ii) repositioning of each center to the centroid of all samples in a given cluster. The algorithm is presented in Fig. 7. As we can see, the common implementation of k-means as a function is non-scalable, because it employs batch model of data transfer: training data are passed as an array of samples, so they must be generated and accumulated in memory before the function is called.

```
function kmeans($data$) returns an array of centers

    Initialize array $centers$
    repeat
        Set $sum[1], \ldots, sum[k], count[1], \ldots, count[k]$ to zero
        for $i = 1..n$ do                /* assign samples to clusters */
            $x = data[i]$
            $j = \text{clusterOf}(x)$
            $sum[j] = sum[j] + x$
            $count[j] = count[j] + 1$
        end
        for $j = 1..k$ do                /* reposition centers */
            $centers[j] = sum[j]/count[j]$
        end
    until no center has been changed
    return $centers$
```

Fig. 7. Pseudocode illustrating k-means clustering algorithm implemented as a regular stand-alone function. The function takes an array of n samples ($data$) as argument and returns k cluster centers. Both samples and centers are real-valued vectors. The function clusterOf(x) returns index of the center that is closest to x.

```
class KMeans extends Cell
method build()

    Initialize array $centers$
    repeat
        Set $sum[1], \ldots, sum[k], count[1], \ldots, count[k]$ to zero
(*)     $source$.open()
        for $i = 1..n$ do
(*)         $x = source$.next()
            $j = \text{clusterOf}(x)$
            $sum[j] = sum[j] + x$
            $count[j] = count[j] + 1$
        end
(*)     $source$.close()
        for $j = 1..k$ do
            $centers[j] = sum[j]/count[j]$
        end
    until no center has been changed
```

Fig. 8. Pseudocode illustrating implementation of k-means as Debellor's cell. Since k-means is a training algorithm (generates a decision model), it must be implemented in method build() of a Cell's subclass. Input data are provided by the *source* cell, the reference *source* being a field of Cell. The generated model is stored in the field *centers* of class KMeans, method build() does not return anything. The lines of code inserted or modified relatively to the standard implementation are marked with asterisk (*).

```
class KMeans extends Cell
method next()
    $x = source.next()$
    if $x$ == null then
        return null
    return $x$.setLabel(clusterOf($x$))
```

Fig. 9. Pseudocode illustrating implementation of method next() of KMeans cell. This method employs the clustering model generated by build() and stored inside the KMeans object to label new samples with identifiers of their clusters.

```
/* 3 cells are created and linked into DPN */
Cell arff = new ArffReader();
arff.set("filename", "iris.arff");        /* parameter filename is set */

Cell remove = new WekaFilter("attribute.Remove");
remove.set("attributeIndices", "last");
remove.setSource(arff);                   /* cells arff and remove are linked */

Cell kmeans = new KMeans();
kmeans.set("numClusters", "10");
kmeans.setSource(remove);

/* k-means algorithm is executed */
kmeans.build();

/* the clusterer is used to label 3 training samples with cluster identifiers */
kmeans.open();
Sample s1 = kmeans.next(),
       s2 = kmeans.next(),
       s3 = kmeans.next();
kmeans.close();

/* labelled samples are printed on screen */
System.out.println(s1 + "\n" + s2 + "\n" + s3);
```

Fig. 10. Java code showing sample usage of Debellor cells: reading data from an ARFF file, removal of an attribute, training and application of a k-means clusterer

Stream implementation of k-means – as Debellor's cell – is presented in Fig. 8. In contrast to the standard implementation, training data are *not* passed explicitly, as an array of samples. Instead, the algorithm retrieves samples one-by-one from the source cell, so it can process arbitrarily large data sets. In addition, Fig. 9 shows how to implement method `next()`, responsible for application of the generated clustering model to new samples.

Note that despite the algorithm presented in Fig. 8 employs *stream* method of data transfer, it employs *batch* method of updating the decision model (the updates are performed after all samples have been scanned). These two things – the method of data transfer and the way how model is updated – are separate and independent issues. It is possible for batch (in terms of model update) algorithms to utilize and benefit from stream architecture.

Listing in Fig. 10 shows how to run a simple experiment: train a k-means clusterer and apply it to several training samples, to label them with identifiers of their clusters. Data are read from an ARFF file and simple preprocessing – removal of the last attribute – is applied to all samples. Note that loading data from file and preprocessing is executed only when the next input sample is requested by the `kmeans` cell – in methods `build()` and `next()`.

6 Experimental Evaluation

6.1 Setup

In existing data mining systems, when data to be processed are too large to fit in memory, they must be put in *virtual* memory. During execution of the algorithm, parts of data are being swapped to disk by operating system, to make space for other parts, currently requested. In this way, portions of data are constantly moving between memory and disk, generating huge overhead on execution time of the algorithm. In the presented experiments we wanted to estimate this overhead and the performance gain that can be obtained through the use of Debellor's data streaming instead of swapping.

For this purpose, we trained k-means [17,24,25] clustering algorithm on time windows extracted from the time series that was used in EUNITE[12] 2003 data mining competition. We compared execution times of two variants of the experiment:

1. *batch*, with time windows created in advance and buffered in memory,
2. *stream*, with time windows generated on the fly.

Data Processing Networks of both variants are presented in Fig. 11 and 12.

In both variants, we employed our stream implementation of k-means, sketched in Sect. 5.8 (KMeans cell in Fig. 11 and 12). In the first variant, we inserted a buffer into DPN just before the KMeans cell – in this way we effectively obtained a batch algorithm. In the second variant, the buffer was placed earlier in the chain of algorithms, before window extraction. We could have dropped buffering at all, but then the data would be loaded from disk again in every training cycle, which was not necessary, as the source data were small enough to fit in memory.

[12] EUropean Network on Intelligent TEchnologies for Smart Adaptive Systems, http://www.eunite.org

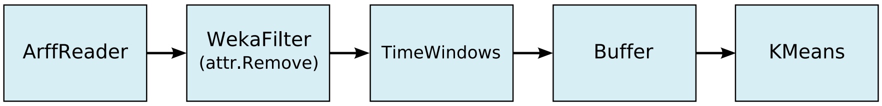


Fig. 11. DPN of the first (batch) variant of experiment

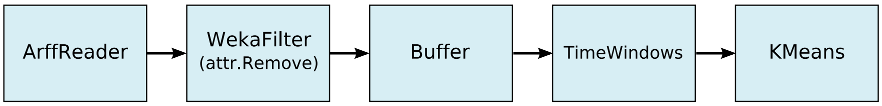


Fig. 12. DPN of the second (stream) variant of experiment

Source data were composed of a series of real-valued measurements from glass production process, recorded in 9408 different time points separated by 15-minute intervals. There were two kinds of measurements: 29 "input" and 5 "output" values. In the experiment we used only "input" values, "output" ones were filtered out by Weka filter for attribute removal (WekaFilter cell).

After loading from disk and dropping unnecessary attributes, the data occupied 5.7MB of memory. They were subsequently passed to TimeWindows cell, which generated time windows of length W, on every possible offset from the beginning of the input time series. Each window was created as a concatenation of W consecutive samples of the series. Therefore, for input series of length T, composed of A attributes, the resulting stream contained $T - W + 1$ samples, each composed of $W * A$ attributes. In this way, relatively small source data (5.7MB) generated large volume of data at further stages of DPN, e.g. 259MB for $W = 50$.

In the experiments, we compared training times of both variants of k-means. Since the time effectiveness of swapping and memory management depends highly on the hardware setup, the experiments were repeated in two different hardware environments: (A) a laptop PC with Intel Mobile Celeron 1.7 GHz CPU, 256MB RAM; (B) a desktop PC with AMD Athlon XP 2100+ (1.74 GHz), 1GB RAM. Both systems run under Microsoft Windows XP. Sun's Java Virtual Machine (JVM) 1.6.0_03 was used. The number of clusters for k-means was set to 5.

6.2 Results

Results of experiments are presented in Table 1 and 2. They are also depicted graphically in Fig. 13 and 14.

Different lengths of time windows were checked, for every length the size of generated training data was different (given in the second column of the tables). In each trial, training time of k-means was measured. These times are reported in normalized form, i.e. the total training time in seconds is divided by the number of training cycles and data size in MB. Normalized times can be directly

Table 1. Normalized training times of k-means for batch and stream variant of experiment and different lengths of time windows. Corresponding sizes of training data are given in the second column. Hardware environment A.

Window length	Data size [MB]	Normalized execution time (batch variant)	Normalized execution time (stream variant)
10	53	3.1	5.6
20	104	3.2	5.3
30	156	3.1	5.0
40	208	5.1	4.9
50	259	244.4	5.0
60	311	326.9	8.3
70	362	370.6	10.7
80	413	386.0	10.9
90	464	475.3	11.1

Table 2. Normalized training times of k-means for batch and stream variant of experiment and different lengths of time windows. Corresponding sizes of training data are given in the second column. Hardware environment B.

Window length	Data size [MB]	Normalized execution time (batch variant)	Normalized execution time (stream variant)
50	259	4.0	5.3
100	515	4.0	5.4
120	617	4.0	6.5
150	769	5.3	8.7
170	869	6.3	8.8
180	919	23.8	8.8
190	969	36.4	8.8
200	1019	50.7	8.8
210	1069	71.3	8.8
220	1119	85.1	8.8
230	1168	100.4	9.1
240	1218	111.1	9.1
250	1267	140.2	9.4
260	1317	**crash**	9.3

compared across different trials. Every table and figure presents results of both variants of the algorithm.

Time complexity of a single training cycle of k-means is linear in the data size, so normalized execution times should be similar across different values of window length. However, for the batch variant, the times are constant only for small sizes of data. At the point when data size gets close to the amount of physical memory installed on the system, execution time suddenly jumps to a very high value, many times larger than for smaller data sizes. It may even

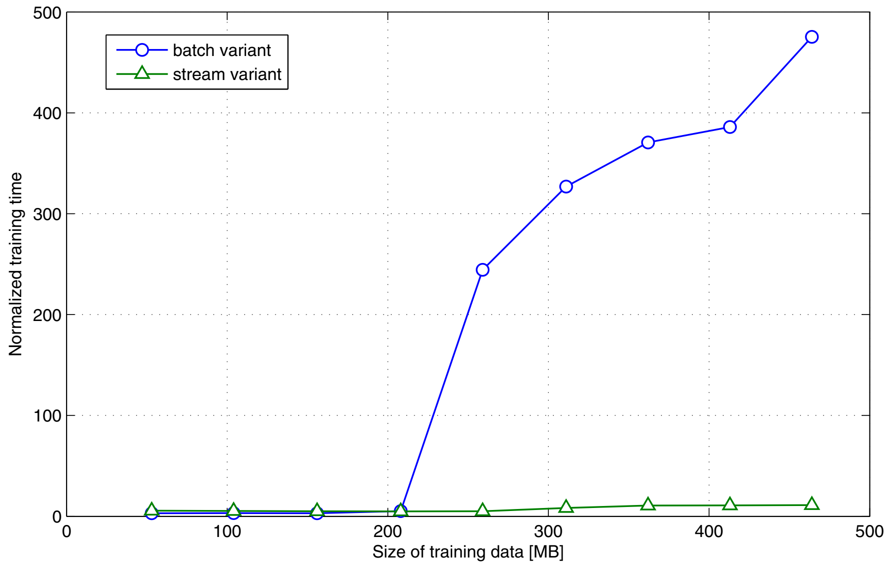


Fig. 13. Normalized training times of k-means for batch and stream variant of experiment and different lengths of time windows. Hardware environment A.

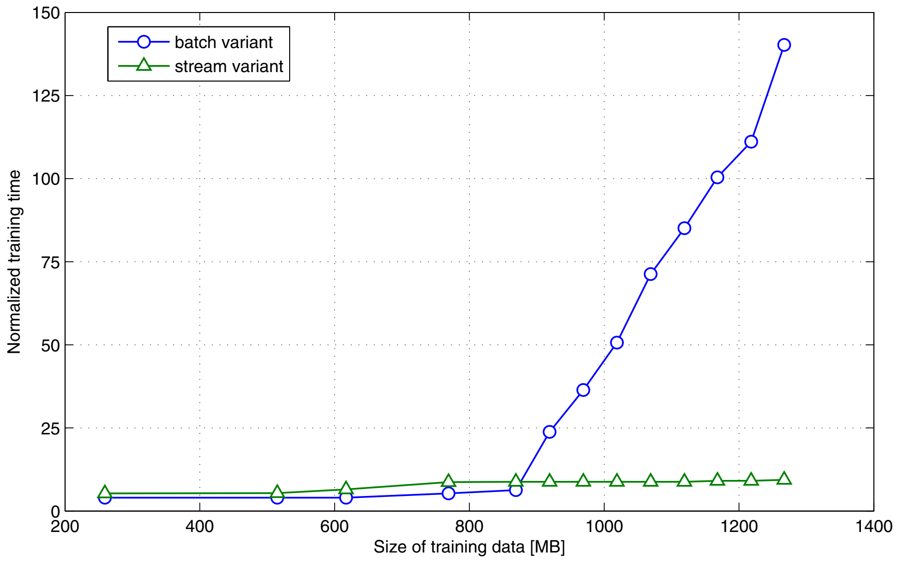


Fig. 14. Normalized training times of k-means for batch and stream variant of experiment and different lengths of time windows. Hardware environment B. Note that the measurement which caused the batch variant to crash (last row in Table 2) is not presented here.

happen that from some point the execution crashes due to memory shortage (see Tab. 2), despite JVM heap size being set to the highest possible value (1300 MB on a 32-bit system). This is because swapping must be activated to handle this large volume of data. And because access to disk is orders of magnitude slower than to memory, algorithm execution becomes also very slow.

This dramatic slowdown is not present in the case of the stream algorithm, which requires always the same amount of memory, at the level of 6MB. For small data sizes this algorithm runs a bit slower, because training data must be generated in each training cycle from the beginning. But for large data sizes it can be 40 times better, or even more (the curves in Figures 13 and 14 rise very quickly, so we may suspect that for larger data sizes the disparity between both variants is even bigger). The batch variant is actually not usable.

What is also important, every stream implementation of a data mining algorithm can be used in batch manner by simply preceding it with a buffer in DPN. Thus, the user can choose the faster variant, depending on the data size. On the other hand, batch implementation *cannot* be used in stream-based manner, rather the algorithm must be redesigned and implemented again.

7 Conclusions

In this paper we introduced Debellor – a data mining platform with stream architecture. We presented the concept of data streaming and proved through experimental evaluation that it enables much more efficient processing of large data than the currently used method of batch data transfer. Stream architecture is also more general. Every stream-based implementation can be used in batch manner. Opposite is not true. Thanks to data streaming, algorithms implemented on Debellor platform can be scalable and interoperable at the same time. We also analysed the significance of scalability issue for the design of composite data mining systems and showed that even when source data are relatively small, lack of memory may still pose a problem, since large volumes of data may be generated at intermediate stages of data processing network.

Stream architecture has also weaknesses. Because of sequential access to data, implementation of algorithms may be conceptually more difficult. Batch data transfer is more intuitive for the programmer. Moreover, some algorithms may inherently require random access to data. Although they can be implemented in stream architecture, they have to buffer all data internally, so they will not benefit from streaming. However, these algorithms can still benefit from interoperability provided by Debellor – they can be connected with other algorithms to form a complex data mining system.

Development of Debellor will be continued. We plan to extend the architecture to handle multi-input and multi-output cells as well as nesting of cells (e.g., to implement meta-learning algorithms). We also want to implement parallel execution of DPN and serialization of cells (i.e., saving to a file).

Acknowledgement. The research has been partially supported by the grant N N516 368334 from Ministry of Science and Higher Education of the Republic of Poland and by the grant “Decision support – new generation systems” of Innovative Economy Operational Programme 2008-2012 (Priority Axis 1. Research and development of new technologies) managed by Ministry of Regional Development of the Republic of Poland.

References

1. Aggarwal, C.C. (ed.): Data Streams: Models and Algorithms. Springer, Heidelberg (2007)
2. Gama, J., Gaber, M.M. (eds.): Learning from Data Streams: Processing Techniques in Sensor Networks. Springer, Heidelberg (2007)
3. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Prentice Hall, Englewood Cliffs (2002)
4. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. IEEE Computer Vision and Pattern Recognition 1, 511–518 (2001)
5. Bazan, J.G., Szczuka, M.: RSES and RSESlib – A collection of tools for rough set computations. In: Ziarko, W.P., Yao, Y. (eds.) RSCTC 2000. LNCS, vol. 2005, pp. 106–113. Springer, Heidelberg (2001)
6. Bazan, J.G., Szczuka, M.S., Wojna, A., Wojnarski, M.: On the evolution of rough set exploration system. In: Tsumoto, S., Słowiński, R., Komorowski, J., Grzymała-Busse, J.W. (eds.) RSCTC 2004. LNCS, vol. 3066, pp. 592–601. Springer, Heidelberg (2004)
7. Wojna, A., Kowalski, L.: Rseslib: Programmer’s Guide (2008), `http://rsproject.mimuw.edu.pl`
8. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
9. R Development Core Team: R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria (2005)
10. Eaton, J.W.: Octave: Past, present, and future. In: International Workshop on Distributed Statistical Computing (2001)
11. Eaton, J.W., Rawlings, J.B.: Ten years of Octave – recent developments and plans for the future. In: International Workshop on Distributed Statistical Computing (2003)
12. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: Yale: Rapid prototyping for complex data mining tasks. In: KDD 2006: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (2006)
13. Garcia-Molina, H., Ullman, J.D., Widom, J.: Database System Implementation. Prentice Hall, Englewood Cliffs (1999)
14. Garcia-Molina, H., Ullman, J.D., Widom, J.: Database Systems: The Complete Book. Prentice Hall, Englewood Cliffs (2001)
15. Arasu, A., Babcock, B., Babu, S., Cieslewicz, J., Datar, M., Ito, K., Motwani, R., Srivastava, U., Widom, J.: STREAM: The stanford data stream management system (2004), `http://dbpubs.stanford.edu:8090/pub/2004-20`
16. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: ACM (ed.) Symposium on Principles of Database Systems, pp. 1–16. ACM Press, New York (2002)
17. Ripley, B.D.: Pattern recognition and neural networks. Cambridge University Press, Cambridge (1996)

18. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)
19. Bradley, P.S., Fayyad, U.M., Reina, C.: Scaling clustering algorithms to large databases. In: Knowledge Discovery and Data Mining (1998)
20. Balakrishnan, S., Madigan, D.: Algorithms for sparse linear classifiers in the massive data setting. Journal of Machine Learning Research 9, 313–337 (2008)
21. Amit, Y., Shalev-Shwartz, S., Singer, Y.: Online learning of complex prediction problems using simultaneous projections. Journal of Machine Learning Research 9, 1399–1435 (2008)
22. Furaoa, S., Hasegawa, O.: An incremental network for on-line unsupervised classification and topology learning. Neural Networks 19, 90–106 (2006)
23. Kivinen, J., Smola, A.J., Williamson, R.C.: Online learning with kernels. IEEE Transactions On Signal Processing 52, 2165–2176 (2004)
24. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys 31, 264–323 (1999)
25. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, Englewood Cliffs (1995)
26. Wojnarski, M.: Absolute contrasts in face detection with adaBoost cascade. In: Yao, J., Lingras, P., Wu, W.-Z., Szczuka, M.S., Cercone, N.J., Ślęzak, D. (eds.) RSKT 2007. LNCS, vol. 4481, pp. 174–180. Springer, Heidelberg (2007)
27. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324 (1998)
28. Kriegel, H.P., Borgwardt, K.M., Kröger, P., Pryakhin, A., Schubert, M., Zimek, A.: Future trends in data mining. Data Mining and Knowledge Discovery 15, 87–97 (2007)
29. Pyle, D.: Data Preparation for Data Mining. Morgan Kaufmann Publishers, San Francisco (1999)
30. Booch, G., Rumbaugh, J., Jacobson, I.: Unified Modeling Language User Guide. Addison-Wesley, Reading (2005)
31. Silberschatz, A., Galvin, P., Gagne, G.: Operating System Concepts, 7th edn. Wiley, Chichester (2004)

Category-Based Inductive Reasoning: Rough Set Theoretic Approach

Marcin Wolski

Department of Logic and Methodology of Science,
Maria Curie-Skłodowska University, Poland
marcin.wolski@umcs.lublin.pl

Abstract. The present paper is concerned with rough set theory (RST) and a particular approach to human-like induction, namely similarity coverage model (SCM). It redefines basic concepts of RST – such like e.g. a decision rule, accuracy and coverage of decision rules – in the light of SCM and explains how RST may be viewed as a similarity-based model of human-like inductive reasoning. Furthermore, following the knowledge-based theory of induction, we enrich RST by the concept of an ontology and, in consequence, we present an RST-driven conceptualisation of SCM. The paper also discusses a topological representation of information systems in terms of non-Archimedean structures. It allows us to present an ontology-driven interpretation of finite non-Archimedean nearness spaces and, to some extent, to complete recent papers about RST and the topological concepts of nearness.

1 Introduction

Category-based induction is an approach to human-like inductive reasoning in which both conceptual knowledge and similarity of objects play the key role. So far this type of reasoning has been a subject of study mainly in ethnobiology, or better still, in cognitive science. In this paper we shall apply the main ideas underlying category-based induction to computer science, especially to rough set theory (RST) [10,12]. It will allow us to introduce some new interesting interpretations of basic concepts and structures from RST and topology.

There are, in general, two basic theories explaining the mechanism of (human-like) induction: the knowledge-based theory and the similarity-based theory. According to the former one, induction is driven by a prior categorisation of given objects, often called conceptual knowledge or ontology. On this view, people firstly identify some category of which a given object is an element and then generalise properties to the members of this category and vice versa. For example, knowing that bluejays require vitamin K for their liver to function, one can generalise that all birds require this vitamin too. On the other hand, the similarity-based theory argues that induction is based on the overall similarity of compared objects rather than on the conceptual knowledge. For example, students from Michigan are reported to conclude – on the basis that skunks have some biological property – that it is more likely that rather opossums have this property than bears. Skunks, however, are taxonomically closer to bears than to opossums [2]. Summing up, according to the knowledge-based approach the generalisation from one object to another is supported by categories (agents ignore appearances of objects and rely on the category membership), whereas according to the similarity-based approach such generalisations are based on perceptual similarity (agents ignore their knowledge about the category membership and rely on appearances) [7].

Inductive reasoning which takes into account both conceptual knowledge and similarity of objects is generally called category-based induction. There are a number of formal models of such reasoning, e.g. [2,6,9,15]. In this paper we shall study similarity coverage model (SCM) [2,9], mainly for its simplicity and strong influence on the development of other models. According to SCM, the strength of inductive argument increases with (a) the degree to which the premies categories are similar to the conclusion category, and (b) the degree to which premise categories cover the lowest level knowledge-category (e.g. from a taxonomy) that includes both the premies and conclusion categories. Thus, the step (a) represents the similarity-based approach, whereas the step (b) represents the knowledge-based approach.

The main aim of this paper is to give an account of SCM within the conceptual framework of RST. First, we re-interpret some notions from RST – such as a decision rule, accuracy and coverage of decision rules, rough inclusion functions – according to the standpoint of SCM. On this view, RST may be regarded as a similarity-based approach to induction. Then we enrich RST with a proper ontology and show how the knowledge-based approach can correct the assessment of decision rules. In consequence, the paper proposes an RST-driven model of category-based induction. Furthermore, we discuss topological aspects of RST and the category-based theory of induction. More specifically, we examine a topological counterpart of an information system. Usually, information systems have been represented as approximation spaces or approximation topological spaces. In consequence, only the indiscernability (or similarity) relation induced by the set of all attributes has been considered. In contrast to this approach, we consider all indiscernability relations, or better still, all partitions, induced by an information system. Mathematically, these partitions induce a non-Archimedean structure which, in turn, gives rise to a topological nearness space. Recently a number of attempts have been made to connect RST with nearness type structures, e.g. [11,19]. To some extent we complete the previous results and show some intuitive reasons to consider such structures. Specifically, every ontology induced over a non-Archimedean structure is taxonomic.

The paper is organised as follows. Section 2 contains a brief and informal introduction to SCM. Section 3 describes basic concepts from RST which are relevant to inductive reasoning. Section 4 discusses concepts introduced in Section 3 against the background of inductive reasoning and SCM. Finally, Section 5 presents topological aspects of RST and the category-based approach to induction.

2 Similarity Coverage Model

In this section we informally introduce category-based induction which has been of a special importance mainly for ethnobiology. There are different accounts of such induction – in the paper we focus on the very influential similarity coverage model (SCM) introduced by Osherson et al. [9].

Ethnobiology or folk biology is a branch of cognitive science which studies the ways in which people categorise the local fauna and flora and project their knowledge about a certain category to other ones [2,6,9,15]. For example, given that *bobcats secrete uric acid crystals* and *cows secrete uric acid crystals*, subjects, on the basis that all mammals may have this property, infer that *foxes secrete uric acid crystals*.

According to SCM, the subject performing an induction task firstly calculates the *similarity* of the premise categories (i.e. bobcats, cows) to the conclusion category (i.e. foxes). Then the subject calculates the average similarity (*coverage*) of the premise categories to the superordinate category including both the premise and conclusion categories (i.e. mammals). Let us consider the following example:

$$\begin{array}{c}\textit{Horses have an ileal vein,}\\ \textit{Donkeys have an ileal vein.}\\ \hline \textit{Gophers have an ileal vein.}\end{array}$$

This argument is weaker than:

$$\begin{array}{c}\textit{Horses have an ileal vein,}\\ \textit{Gophers have an ileal vein.}\\ \hline \textit{Cows have an ileal vein.}\end{array}$$

Of course, the similarity of horses to cows is much higher than the similarity of horses or donkeys to gophers. Thus the strength of inductive inference depends on the maximal similarity of the conclusion category to some of the premise categories. Now let us shed some light on the coverage principle:

$$\begin{array}{c}\textit{Horses have an ileal vein,}\\ \textit{Cows have an ileal vein.}\\ \hline \textit{All mammals have an ileal vein.}\end{array}$$

According to SCM this argument is weaker than the following one:

$$\begin{array}{c}\textit{Horses have an ileal vein,}\\ \textit{Gophers have an ileal vein.}\\ \hline \textit{All mammals have an ileal vein.}\end{array}$$

The reason is that the average similarity of horses to other mammals is almost the same as that of cows. In other words, the set H of all animals considered to be similar to horses is almost equal to the set C of all animals similar to cows. Thus the second premise does not bring us nothing in terms of coverage. By contrast, gophers are similar to other mammals than horses and thus this premise makes the coverage higher. That is, the set $H \cup G$, where G is the set of all animals similar to gophers, has more elements than the set $H \cup C$. Thus, the following inductive inference

$$\begin{array}{c}\textit{Horses have an ileal vein,}\\ \hline \textit{All mammals have an ileal vein.}\end{array}$$

is stronger, than

$$\begin{array}{c}\textit{Bats have an ileal vein,}\\ \hline \textit{All mammals have an ileal vein.}\end{array}$$

The range of mammals similar to cows is much wider than the range of mammals similar to bats. One can say that cows are more typical examples of mammals than bats or gophers.

Now, let us summarise the above examples in a more formal way. Firstly, there is given a set of categories $\mathcal{C}$ we reason about. This set is provided with a binary "kind of" relation K, which is acyclic and thus irreflexive and asymmetric. We call K *taxonomic* if and only if it is transitive and no item is of two distinct kinds.

Definition 1. *A transitive relation K is taxonomic over C iff for any $a, b, c \in C$ such that aKb and aKc, it holds that $b = c$ or bKc or cKb.*

For example, *collie* is a kind of *dog* and *dog* is a kind of *mammal*. Items $x \in C$, such that there is no t satisfying tKx constitute basic categories. An example of non-taxonomic relation is as follows: *wheelchair* is a kind of *furniture* and a kind of *vehicle*. Now, neither *furniture* = *vehicle* nor *furnitureKvehicle* nor *vehicleKfurniture*.

Subjects reasoning about $\mathcal{C}$ are additionally provided with a default notion of similarity R defined on basic categories $\mathcal{C}_{Basic}$, i.e. minimal elements of $\mathcal{C}$ with respect to K. People usually assume that R is at least reflexive and symmetric. Very often R is represented as an equivalence relation, that is, R is additionally transitive. Given that $c_1 \in \mathcal{C}_{Basic}$ has a property p, a subject may infer that $c_2 \in \mathcal{C}_{Basic}$ also satisfies p, if there exists $c_3 \in \mathcal{C}$ such that c_1Kc_3, c_2Kc_3 and $\{c \in \mathcal{C}_{Basic} : c_1Rc\}$ is a "substantial" subset of $\{c \in \mathcal{C}_{Basic} : cKc_3\}$. Informally speaking, one can transfer knowledge from a category c_1 to c_2 if the set of all elements considered to be similar to c_1 is a substantial subset of the set of all $\mathcal{C}_{basic}$-instantiations of the minimal taxonomic category c whose examples are c_1 and c_2. Summing up, one can say that $(\mathcal{C}, K)$ represents gathered information, while R is an inductive "engine" making inferences about unknown features of objects belonging to $\mathcal{C}$.

3 Rough Set Theory

In this section we briefly recall basic notions from RST which are relevant to inductive reasoning. We start with introducing the concept of an information system, then we discuss decision rules and different measures of their strength. We conclude by recalling some notions from the rough–mereological approach.

Definition 2. *An information system is a quadruple $\langle U, A, V, f\rangle$ where:*

- *U is a non–empty finite set of objects;*
- *A is a non–empty finite set of attributes;*
- *$V = \bigcup_{a \in A} V_a$ where V_a is the value–domain of the attribute a;*
- *$f : U \times A \mapsto V$ is an information function, such that for all $a \in A$ and $u \in U$, $f(u, a) \in V_a$.*

It is often useful to view an information system $\langle U, A, V, f\rangle$ as a decision table, assuming that $A = C \cup D$ and $C \cap D = \emptyset$ where C is a set of conditional attributes and D is a set of decision attributes. For example, Figure 1 presents a decision table where:

- $U = \{Beaver, Squirrel, Mouse, Muskrat, Otter, Skunk\}$,
- $C = \{Environment, Diet, Tail, Size\}$,

Animals	Environment	Diet	Tail	Size	Poland
Beaver	semi-aquatic	herbivorous	flattened	medium	yes
Squirrel	terrestial	omnivorous	round	small	yes
Mouse	terrestial	omnivorous	round	very small	yes
Muskrat	semi-aquatic	omnivorous	round	medium	yes
Otter	semi-aquatic	carnivorous	round	medium	yes
Skunk	terrestial	omnivorous	round	medium	no

Fig. 1. An example of a dataset

- $D = \{Poland\}$,
- e.g. $V_{Diet} = \{herbivorous, carnivorous, omnivorous\}$ for $Diet \in C$.

Each subset of attributes $S \subseteq A$ determines an equivalence relation $IND(S) \subseteq U \times U$ defined as follows:

$$IND(S) = \{(u, v) : (\forall a \in S)\ f(u, a) = f(v, a)\}.$$

As usual, $IND(S)$ is called an indiscernability relation induced by S, the partition induced by the relation $IND(S)$ is denoted by $U/IND(S)$, and $[u]_S$ denotes the equivalence class of $IND(S)$ defined by $u \in U$. For instance, if $S = \{Environment, Diet\}$, then $IND(S) = \{\{Beaver\}, \{Squirrel, Mouse, Skunk\}, \{Muskrat\}, \{Otter\}\}$.

Obviously, $U/IND(A)$ refines every other partition $U/IND(S)$ where $S \subseteq A$. Furthemore

$$[u]_A = \bigcap_{S \subseteq A} [u]_S.$$

Intuitively, any subset $X \subseteq U$ which can be defined by a formula of some knowledge representation language $\mathcal{L}$ is a concept in $\mathcal{L}$. For example, one can use the following simple descriptor language, say $\mathcal{L}_{Desc}$, based on a given information system $\langle U, A, V, f\rangle$:

$$fml ::= [a = val] \mid \neg fml \mid fml \wedge fml \mid fml \vee fml$$

where $a \in A$ and $val \in V_a$. We say that $\alpha \in \mathcal{L}_{Desc}$ is a formula over C if all attributes a in α belong to C.

For any formula $\alpha \in \mathcal{L}_{Desc}$, $|\alpha|$ denotes the meaning of α in U, i.e. the concept in $\mathcal{L}_{Desc}$ which is defined as follows:

- If α is of the form $[a = val]$, then $|\alpha| = \{u \in U : f(u, a) = val\}$;
- $|\neg\alpha| = U \setminus |\alpha|$, $|\alpha \wedge \beta| = |\alpha| \cap |\beta|$, $|\alpha \vee \beta| = |\alpha| \cup |\beta|$.

For example, $\alpha = [Poland = no]$ and $|\alpha| = \{Skunk\}$.

Let α be a formula of $\mathcal{L}_{Desc}$ over C and β a formula over D. Then the expression $\alpha \Rightarrow \beta$ is called a decision rule if $|\alpha|_A \cap |\beta|_A \neq \emptyset$.

Definition 3. *Let $\alpha \Rightarrow \beta$ be a decision rule and $Card(B)$ denote the cardinality of the set B. Then, the accuracy $Acc_\alpha(\beta)$ and the coverage $Cov_\alpha(\beta)$ for $\alpha \Rightarrow \beta$ are defined as follows:*

$$Acc_\alpha(\beta) = \frac{Card(|\alpha| \cap |\beta|)}{Card(\alpha)},$$

and

$$Cov_\alpha(\beta) = \frac{Card(|\alpha| \cap |\beta|)}{Card(\beta)}.$$

Example 1. Let us assume that $|\alpha| = [Environment = semi - aquatic]$ and $|\beta| = [Poland = yes]$. Then $\alpha \Rightarrow \beta$ is a decision rule over the information system depicted by Fig. 1, $Acc_\alpha(\beta) = 3/3 = 1$, and $Cov_\alpha(\beta) = 3/5$.

It is worth emphasising that if $Acc_\alpha(\beta) = 1$, then it holds that $u \in |\beta|$ provided that $u \in |\alpha|$. On the other hand, if $Cov_\alpha(\beta) = 1$ then we have $u \in |\alpha|$ provided that $u \in |\beta|$. Thus, $Acc_\alpha(\beta)$ measures the sufficiency of $\alpha \Rightarrow \beta$, whereas $Cov_\alpha(\beta)$ measures the necessity of $\alpha \Rightarrow \beta$; for details see e.g. [16]. Hereafter several attempts were made to introduce other measures of how good a given decision rule is. However, the meaning of these measures remains fixed. For a given decision rule $\alpha \Rightarrow \beta$ they answer the following question:

$$\text{Given that } u \in |\alpha|, \text{ what is the chance that } u \in |\beta|? \tag{1}$$

In the evolution of RST it is the rough–mereological approach of a special importance [13]. This approach is based on the inclusion function, called rough inclusion (RIF), which generalises fuzzy set and rough set approaches. Generally speaking, RIFs measure the degree of inclusion of a set of objects X in a set of objects Y. In this paper we follow the definition of RIF proposed in [5]:

Definition 4. *A RIF upon U is any function $\kappa : 2^U \times 2^U \mapsto [0,1]$ such that:*

- $(\forall X, Y)(\kappa(X,Y) = 1 \Leftrightarrow X \subseteq Y)$,
- $(\forall X, Y, Z)(Y \subseteq Z \Rightarrow \kappa(X,Y) \leq \kappa(X,Z))$.

The most famous RIF is the so-called standard RIF, denoted by $\kappa^{£}$, which is based on J. Łukasiewicz's ideas concerning the probability of truth of propositional formulas:

$$\kappa^{£}(X,Y) = \begin{cases} \frac{Card(X \cap Y)}{Card(X)} & \text{if } X \neq \emptyset \\ 1 & \text{otherwise} \end{cases}$$

Another RIF κ_1, which is really interesting in the context of induction, was proposed by A. Gomolińska in [5]:

$$\kappa_1(X,Y) = \begin{cases} \frac{Card(Y)}{Card(X \cup Y)} & \text{if } X \cup Y \neq \emptyset \\ 1 & \text{otherwise} \end{cases}$$

As one can easy observe, for any $X, Y \subseteq U$ and any decision rule $\alpha \Rightarrow \beta$,

$$\kappa_1(X,Y) = \kappa^{£}(X \cup Y, Y),$$

$$Acc_\alpha(\beta) = \kappa^{£}(|\alpha|, |\beta|),$$

and

$$Cov_\alpha(\beta) = \kappa^{\pounds}(|\beta|, |\alpha|).$$

Summing up, many ideas from RST are based upon the notion of RIF. In what follows, we shall be interested in question how RIFs can be used to assess the strength of decision rules and their generalisations. It is worth noting that we view these rules from the perspective of inductive reasoning and, in consequence, we change their standard interpretations.

4 Inductive Reasoning: RST Approach

Now, let us examine the above ideas from RST against the background of SCM. As said earlier, each formula $\alpha \in \mathcal{L}_{Desc}$ represents a concept in $\mathcal{L}_{Desc}$. It is easy to observe that

$$|[a = val]| \in U/IND(\{a\}),$$

and

$$|\alpha| = \bigcup \mathcal{A}, \text{ for some } \mathcal{A} \subseteq U/IND(A).$$

Thus, elements of $U/IND(A)$ can be regarded as atomic concepts and any other concept in $\mathcal{L}_{Desc}$ can be built by means of atomic concepts and $\cup$. Furthermore, any formula α will be regarded as a concept name, or better still, a category. Generally speaking, SCM tries to answer the question how safe is to transfer knowledge about a value val of some attribute a from one category α to another category β. In other words,

$$\text{given that } |\alpha| \subseteq |[a = val]|, \text{ what is the chance that } |\beta| \subseteq |[a = val]|? \tag{2}$$

Observe that this question, in contrast to (1), makes sense even for a rule $\alpha \Rightarrow \beta$ such that $|\alpha| \cap |\beta| = \emptyset$. We shall call such a rule an inductive rule. Furthermore, examples from Section 2 require multi-premises inductive rules represented by expressions $\alpha, \beta, \gamma \Rightarrow \delta$ rather than simple rules of the form $\alpha \Rightarrow \delta$. Let us recall that in multiconclusion Gentzen's sequent calculus

$$\alpha, \beta, \gamma \Rightarrow \delta \text{ means that } \delta \text{ follows from } \alpha \wedge \beta \wedge \gamma.$$

However, in SCM-like inductive reasoning we have

$$\alpha, \beta, \gamma \Rightarrow \delta \text{ means that } \delta \text{ follows from } \alpha \vee \beta \vee \gamma. \tag{3}$$

Indeed, for example the following decision rule

$$[Size = verysmall], [Size = small] \Rightarrow [Poland = yes]$$

based on the dataset from Fig. 1 where

$$|[Size = verysmall]| = \{Mouse\},$$

$$|[Size = small]| = \{Squirrel\},$$

$$|[Poland = yes]| = \{Beaver, Squirrel, Mouse, Muskrat, Otter\}$$

might represent the following inductive inference:

$$\frac{\begin{array}{c}\textit{Mice have an ileal vein,}\\ \textit{Squirrels have an ileal vein,}\end{array}}{\textit{All animals living in Poland have an ileal vein.}}$$

As one can easy observe $|[Size = verysmall] \wedge [Size = small]| = \emptyset$ and the conjunctive interpretation of premises leads us to wrong conclusions. Thus, given the rule $\alpha, \beta, \gamma \Rightarrow \delta$ we shall regard the premises as the category $\alpha \vee \beta \vee \gamma$ representing the concept $|\alpha \vee \beta \vee \gamma|$ in $\mathcal{L}_{Desc}$.

Now, we answer the question (2). According to SCM we should (a) compute the similarity of the premise category to the conclusion category, and (b) compute the degree to which the premise category cover the lowest level knowledge–category that includes both the premies and conclusion categories.

Intuitively, the identity is the highest level of similarity. It is easy to observe that

$$|\alpha| = |\beta| \text{ iff } Acc_\alpha(\beta) = 1 \text{ and } Cov_\alpha(\beta) = 1.$$

Thus, the measures $Acc_\alpha(\beta)$ and $Cov_\alpha(\beta)$ taken together tell us to which extent the categories $|\alpha|$ and $|\beta|$ are similar. In this paper we use the following measure, denoted by Sim, which was firstly introduced by S. Kulczyński in the context of clustering methods [8]; see also [1]:

$$Sim(\alpha, \beta) = \frac{Acc_\alpha(\beta) + Cov_\alpha(\beta)}{2}.$$

It is easy to compute that for $\alpha \Rightarrow \beta$ from Example 1 $Sim(\alpha, \beta) = 4/5$. Now, let us consider the step (b) of SCM. It can be formalised as follows:

$$Cov(\alpha, \beta) = \frac{Card(|\alpha|)}{Card(\mathcal{C})}$$

where $\alpha \Rightarrow \beta$ is an inductive rule, and $\mathcal{C}$ represents the smallest category from the underlying ontology $\mathcal{O}$ containing both the premise and conclusion categories, i.e. $|\alpha| \cup |\beta| \subseteq |\mathcal{C}|$. Since RST assumes that any formula $\alpha \in \mathcal{L}_{Desc}$ representing a concept $|\alpha|$ in $\mathcal{L}_{Desc}$ is a category, the smallest category containing both α and β is $\alpha \vee \beta$. In other words, RST assumes the richest possible ontology representing all concepts definable in $\mathcal{L}_{Desc}$. Thus, we have

$$Cov_{RST}(\alpha, \beta) = \frac{Card(|\alpha|)}{Card(|\alpha| \cup |\beta|)}.$$

Observe that for $\alpha \Rightarrow \beta$ from Example 1,

$$Cov_{RST}(\alpha, \beta) = \kappa_1(\beta, \alpha).$$

Thus, assessing the strength of a rule $\alpha \Rightarrow \beta$ consists in computing values of a couple of RIFs. In our example, for the decision rule $\alpha \Rightarrow \beta$ defined as above, we have

$$Cov_{RST}(\alpha, \beta) = 3/5$$

Some comments may be useful here. The standard RIF $\kappa^{£}$ is the most popular RIF among rough-set community. The main reason seems to be the clarity of the interpretation of $\kappa^{£}$. On the other hand, such RIFs like κ_1 or κ_2 lack an obvious interpretation. Our present inquiry into RST against the background of SCM provide us with the intuitive meaning at least for κ_1: it computes the coverage of the premise category to the smallest category containing both the premise and the conclusion categories (with respect to the ontology representing all concepts in $\mathcal{L}_{Desc}$). It is quite likely that other models of inductive reasoning may bring us some new interpretations of other RIFs as well.

Let us now return to issues concerning the underlying ontology $\mathcal{O}$. First, RST assumes that the ontology $\mathcal{O}$ represents all concepts in $\mathcal{L}_{Desc}$ which, in turn, belong to some $U/IND(S)$ for $S \subseteq A$. That is, atomic concepts are indiscernability classes and all other concepts are built up from them. Second, $\mathcal{O}$ is in fact the Boolean algebra $\langle \mathcal{U}, \cup, \cap, \setminus \rangle$ generated by $U/IND(A)$. Thus, what actually contributes to induction is the indiscernability relation $IND(A)$: when you know $IND(A)$, you also know the corresponding RST ontology $\mathcal{O}$. On this view, RST may be regarded as a kind of the similarity–based approach to inductive reasoning. Only similarity (indiscernability) classes affect induction and, in consequence, only (a) step (i.e. the similarity–based step) of SCM is performed. Since (b) step of SCM assumes additional conceptual knowledge, applying it to RST ontology, which actually brings us nothing more than $IND(A)$, may lead to wrong results.

Example 2. Consider the information system given by Fig. 1. Let

$$\alpha = [Environment = terrestial] \wedge [Diet = omnivorous] \wedge [Tail = round]$$

and

$$\beta = [Poland = no], \neg\beta = [Poland = yes]$$

Then $\alpha \Rightarrow \beta$ is a decision rule and

$$Acc_\alpha(\beta) = 1/3, Cov_\alpha(\beta) = 1/1 = 1,$$

$$Sim(\alpha, \beta) = 2/3, \text{ and } Cov_{RST}(\alpha, \beta) = 3/3 = 1.$$

Also $\alpha \Rightarrow \neg\beta$ is a decision rule, for which we have

$$Acc_\alpha(\neg\beta) = 2/3, Cov_\alpha(\gamma) = 2/5,$$

$$Sim(\alpha, \gamma) = 8/15, \text{ and } Cov_{RST}(\alpha, \beta) = 3/6$$

Observe, that according to the above measures, the arguments represented by $\alpha \Rightarrow \beta$ is stronger than $\alpha \Rightarrow \neg\beta$ (50/30 and 31/30 respectively). However, our intuition suggests us the opposite ranking, e.g.:

Skunks have a property P
Mice have a property P
Squirrels have a property P

All animals not living in Poland have a property P.

Skunks have a property P
Mice have a property P
Squirrels have a property P

All animals in Poland have a property P.

Given that mice and squirrels live in Poland, but skunks do not, it is obvious that the second argument should be recognised as stronger than the first one. In order to correct our result we have to consider a proper ontology $\mathcal{O}$, which brings us new information about U.

Let us look at the scientific ontology given by Fig. 2, which is built for the dataset Fig. 1 – in fact, it is a fragment of the well-known biological taxonomy. In this case, the smallest concept containing both $|\alpha|$ and $|\beta|$ is the set of all objects from the dataset. Thus, $Cov(\alpha, \beta) = 1/2$ and the overall result for $\alpha \Rightarrow \beta$ is $7/6$. The same computation for $\alpha \Rightarrow \neg\beta$ brings us $Cov(\alpha, \neg\beta) = 1/2$ and the overall result is $31/30$. This time the strength of both arguments is quite similar: the difference is equal $4/30$ in favour of the first argument. Thus, this time we have obtained a better result. Our example shows also that all categories used in induction should have proper extensions in the given dataset. For instance, the categories *not living in Poland* and *skunk* represent the same concept $\{Skunk\}$, what actually makes the first argument stronger than the second one even when applying scientific ontology. Observe also that in the case of the scientific ontology beavers and squirrels belong to the same family, yet they differ on all conditional attributes. Thus, this ontology really brings us the new knowledge about the dataset. However, sometimes it is better to have an ontology which reflects our knowledge which is encoded by means of attributes from A. For example, the taxonomy Fig. 3 represents the way people could categorise the animals from Fig. 1. Which ontology is more useful depends on features we want to reason about. For instance, in enthnobiology it is widely agreed that scientific ontology is better to reason about hidden properties of animals, whereas the common sense ontology is better to reason

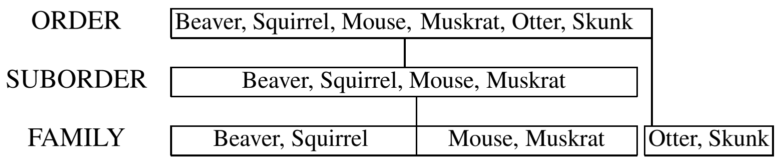

ORDER
Beaver, Squirrel, Mouse, Muskrat, Otter, Skunk
SUBORDER
Beaver, Squirrel, Mouse, Muskrat
FAMILY
Beaver, Squirrel
Mouse, Muskrat
Otter, Skunk

Fig. 2. The scientific taxonomy for the dataset

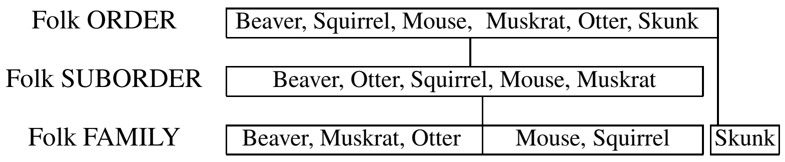

Folk ORDER
Beaver, Squirrel, Mouse, Muskrat, Otter, Skunk
Folk SUBORDER
Beaver, Otter, Squirrel, Mouse, Muskrat
Folk FAMILY
Beaver, Muskrat, Otter
Mouse, Squirrel
Skunk

Fig. 3. A common-sense taxonomy for the dataset

about their behaviour. Thus, the ontology must be carefully chosen with respect to the goal properties.

As said above, the common-sense ontology is mainly based on attributes of objects. On the basis of this fact, one can regard concept lattices from formal concept analysis (FCA) [17,18] as such ontologies. Let us recall that any binary relation $R \subseteq U \times V$ induces two operators:

$$R_+(A) = \{b \in V : (\forall a \in A)\langle a, b\rangle \in R\}$$

$$R^+(B) = \{a \in U : (\forall b \in B)\langle a, b\rangle \in R\}$$

Definition 5. *A* concept *induced by $R \subseteq U \times V$ is a pair (A, B), where $A \subseteq U$ and $B \subseteq V$ such that $A = R^+(B)$ and $B = R_+(A)$. A set A is called an* extent concept *if $A = R^+R_+(A)$. Similarly if $B \subseteq M$ is such that $B = R_+R^+(B)$ then B is called an* intent concept.

The set of all concepts of any information system is a complete lattice [17,18]. Since the lattice induced by our dataset from Fig. 1 is quite complicated, we present here only a list of concepts (see Fig. 4) instead of the Hasse diagram. As one can see, it is quite large ontology when compared with the common sense ontology. As a consequence, for a small dataset as this in the paper the results are very similar to these obtained for RST ontology. However, for large datasets the results may substantially differ. But checking how FCA-ontologies are useful for inductive reasoning is a task for future works.

1. $< \{Beaver\}, \{semiaquatic, herbivorous, flattened, medium, yes\} >$
2. $< \{Squirrel\}, \{terrestial, omnivorous, round, small, yes\} >$
3. $< \{Mouse\}, \{terrestial, omnivorous, round, very\ small, yes\} >$
4. $< \{Muskrat\}, \{semiaquatic, omnivorous, round, medium, yes\} >$
5. $< \{Otter\}, \{semiaquatic, carnivorous, round, medium, yes\} >$
6. $< \{Skunk\}, \{terrestial, omnivorous, round, medium, no\} >$
7. $< \{Beaver, Squirrel, Mouse, Muskrat, Otter\}, \{yes\} >$
8. $< \{Beaver, Muskrat, Otter\}, \{semiaquatic, medium, yes\} >$
9. $< \{Beaver, Muskrat, Otter, Skunk\}, \{medium\} >$
10. $< \{Squirrel, Mouse\}, terrestial, omnivorous, round, yes >$
11. $< \{Squirrel, Mouse, Muskrat\}, \{omnivorous, round, yes\} >$
12. $< \{Squirrel, Mouse, Muskrat, Otter\}, \{round, yes\} >$
13. $< \{Squirrel, Mouse, Skunk\}, \{terrestial, omnivorous, round\} >$
14. $< \{Muskrat, Otter\}, \{semiaquatic, round, medium, yes\} >$
15. $< \{Muskrat, Skunk\}, \{omnivorous, round, medium\} >$
16. $< \{Squirrel, Mouse, Muskrat, Skunk\}, \{omnivorous, round\} >$
17. $< \{Muskrat, Otter, Skunk\}, \{round, medium\} >$
18. $< \{Squirrel, Mouse, Muskrat, Otter, Skunk\}, \{round\} >$
19. $< \{Beaver, Squirrel, Mouse, Muskrat, Otter, Skunk\}, \{\} >$
20. $< \{\}, \{semiaquatic, terrestial, herbivorous, omnivorous,$
    $carnivorous, flattened, round, verysmall, small, medium, yes, no\} >$

Fig. 4. Concepts induced by the dataset from Fig. 1

Summing up this section let us say a few words about inductive rules. As said above, under the interpretation expressed by Equation (2), apart from decision rules also inductive rules make sense. Let $\alpha \in \mathcal{L}_{Desc}$ be a description of beavers, i.e. $|\alpha| = \{Beaver\}$, $\beta \in \mathcal{L}_{Desc}$ be a description of squirrels, $|\beta| = \{Squirrel\}$, and γ be a description of skunks, $|\gamma| = \{Skunk\}$. Then, $|\alpha| \cap |\beta| = \emptyset$, $|\alpha| \cap |\gamma| = \emptyset$, and both $\alpha \Rightarrow \beta$ and $\alpha \Rightarrow \gamma$ are inductive rules. In consequence, $Sim(\alpha, \beta) = Sim(\alpha, \gamma) = 0$. Observe also that Cov_{RST} cannot distinguish these rules either, for we have $Cov_{RST}(\alpha, \beta) = Cov_{RST}(\alpha, \gamma) = 1/2$. However, under the scientific ontology (Fig. 4.) $Cov(\alpha, \beta) = 1/2$, whereas $Cov(\alpha, \gamma) = 1/6$.

5 Induction over Nearness Spaces

In this section we consider a topological counterpart of RST enriched by the concept of ontology. Recently a number of attempts have been made to connect RST with nearness type structures, e.g. [11,19]. These structures (such like nearness spaces or merotopic spaces) are actually quite abstract and this section aims to provide the reader with their intuitive interpretation. We start with some ideas concerning RST and inductive reasoning, and then we develop them into a nearness space.

An information system $\langle U, A, V, f \rangle$ is often represented as an approximation space $(U, IND(A))$, that is, a non-empty set U equipped with an equivalence relation. This representation allows one to connect RST with relational structures which underlie many branches of mathematics, e.g. topology, logic, or universal algebra. Here we would like to change this approach and consider $IND(S)$ for all $S \subseteq A$.

Definition 6. *Let $\mathcal{A}, \mathcal{B} \subseteq 2^X$; then a* refinement relation $\preceq$ *is defined by:*

$$\mathcal{A} \preceq \mathcal{B} \overset{def}{\Leftrightarrow} (\forall A \in \mathcal{A})\ (\exists B \in \mathcal{B}) A \subseteq B.$$

Obviously, for any information system $\langle U, A, V, f \rangle$, $U/IND(A)$ refines every other partition $U/IND(S)$, for all $S \subseteq A$. A simple mathematical structure which generalises this observation is called a *non-Archimedean structure.*

Definition 7. *A non-Archimedean structure μ on a non-empty set U is a set of partitions of U satisfying:*

$$\mathcal{A} \preceq \mathcal{B}\ \&\ \mathcal{A} \in \mu \Rightarrow \mathcal{B} \in \mu,$$

and the couple (U, μ) is called a non-Archimedean space.

Let $\mathcal{IND}_S = \{U/IND(S) : S \subseteq A\}$. Observe that $(U, \mathcal{IND}_S)$ may fail to be a non-Archimedean space. Take as an example the dataset from Fig. 1 and consider the partition $P = \{\{Beaver\}, \{Squirrel, Mouse, Muskrat, Otter, Skunk\}\}$. Then $U/IND(A) \preceq P$, yet there is no $S \subseteq A$ such that $U/IND(S) = P$. Furthermore, any concept $\alpha \in \mathcal{L}_{Desc}$ induces a partition $P_\alpha = \{|\alpha|, |\neg\alpha|\}$ of U and $U/IND(A) \preceq P_\alpha$. For example, when $\alpha = [Diet = herbivorous]$, then $P_\alpha = P$. Thus, what we actually need is a non-Archimedean structure $\mathcal{IND}_A$ induced by $U/IND(A)$:

$$\mathcal{IND}_A = \{P : P \text{ is a partition of } U\ \&\ U/IND(A) \preceq P\}.$$

Proposition 1. *Let $\langle U, A, V, f\rangle$ be an information system, and $\mathcal{C} = \{|\alpha| : \alpha \in \mathcal{L}_{Desc}\}$ be a set of all non-empty concepts in $\mathcal{L}_{Desc}$. Then*

$$\mathcal{C} = \bigcup \mathcal{IND}_A.$$

Proof. We have to prove that $\mathcal{C} \subseteq \bigcup \mathcal{IND}_A$ and $\bigcup \mathcal{IND}_A \subseteq \mathcal{C}$. First, for every non-empty $|\alpha|$ in $\mathcal{L}_{Desc}$ it holds that $U/IND(A) \preceq P_\alpha$ and thus $\mathcal{C} \subseteq \bigcup \mathcal{IND}_A$. Second, assume that $C \in \bigcup \mathcal{IND}_A$. It means that $C \in P$ for some partition $P \in \mathcal{IND}_A$. Since $U/IND(A) \preceq P$, it follows that $C = \bigcup \mathcal{A}$ for some $\mathcal{A} \subseteq U/IND(A)$. Every element C_i of $\mathcal{A}$ is a concept in $\mathcal{L}_{Desc}$ for some $\alpha_i \in \mathcal{L}_{Desc}$ and thus C is a concept in $\mathcal{L}_{Desc}$ for $\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_i$. Hence $\bigcup \mathcal{IND}_A \subseteq \mathcal{C}$.

In other words, all non-empty concepts in $\mathcal{L}_{Desc}$ belong to some partition of the non-Archimedean structure $\mathcal{IND}_A$ and every element of such partition is a concept in $\mathcal{L}_{Desc}$. Since an ontology is a subset of the family of all concepts in $\mathcal{L}_{Desc}$, the space $(U, \mathcal{IND}_A)$ sets the stage for conceptual knowledge about U.

Definition 8. *Let $\langle U, A, V, f\rangle$ be an information system. Then by an* ontology $\mathcal{O}_A$ *over $(U, \mathcal{IND}_A)$ we mean an ordered set of partitions $(\mathcal{P}, \preceq)$ such that $\mathcal{P} \subseteq \mathcal{IND}_A$ and for all $P_i, P_j \in \mathcal{P}$ it holds that $P_i \neq P_j$ for $i \neq j$,*

We say that C is a concept from an ontology $\mathcal{O}_A = (\mathcal{P}, \preceq)$ if $C \in \bigcup \mathcal{P}$. In other words, C is a concept from $\mathcal{O}_A$ if there is a partition $P \in \mathcal{P}$ such that $C \in P$. The set of all concepts from $\mathcal{O}_A$ will be denoted by $\mathcal{C}_{\mathcal{O}_A}$.

Example 3. The scientific taxonomy from Fig. 2 can be represented as follows:

$$\mathcal{O}_A = \{P_1, P_2, P_3, P_4\}$$

where

$$P_1 = \{\{Beaver\}, \{Squirrel\}, \{Mouse\}, \{Muskrat\}, \{Otter\}, \{Skunk\}\},$$
$$P_2 = \{\{Beaver, Squirrel\}, \{Mouse, Muskrat\}, \{Otter, Skunk\}\},$$
$$P_3 = \{\{Beaver, Squirrel, Mouse, Muskrat\}, \{Otter, Skunk\}\},$$
$$P_4 = \{\{Beaver, Squirrel, Mouse, Muskrat, Otter, Skunk\}\}.$$

Definition 9. *We say that C_1 **K** C_2 iff $C_1 \subseteq C_2$ and there exists $P_i, P_j \in \mathcal{O}_A$ such that $C_1 \in P_i, C_2 \in P_j$ and $P_i \preceq P_j$, for all $C_1, C_2 \in \mathcal{C}_{\mathcal{O}_A}$.*

Proposition 2. *The relation **K** is taxonomic over $\mathcal{C}_{\mathcal{O}_A}$ for every ontology $\mathcal{O}_A$ induced the non-Archimedean space $(U, \mathcal{IND}_A)$.*

Proof. It follows from the definition of **K** and the definition of ontology.

For an information system $\langle U, A, V, f\rangle$ the associated taxonomy over $\mathcal{O}_A$ will be denoted by $(\mathcal{C}_{\mathcal{O}_A}, \mathbf{K})$.

In order to generalise this description for non-taxonomic ontologies it suffices to define the ontology over the family of covers.

Definition 10. $stack_{\preceq}\mu = \{\mathcal{B} \subseteq 2^X : (\exists \mathcal{A} \in \mu\mathcal{A}) \preceq \mathcal{B}\}$.

First, $stack_{\preceq}\mathcal{IND}_A$ is a family of covers of U. Second, for an information system $\langle U, A, V, f\rangle$, as a generalised ontology we take an ordered set of covers $(\mathcal{P}, \preceq)$.

Definition 11. *Let $\langle U, A, V, f\rangle$ be an information system. Then by an* generalised ontology $\mathcal{GO}_A$ *over $(U, \mathcal{IND}_A)$ we mean an ordered set of partitions $(\mathcal{P}, \preceq)$ such that $\mathcal{P} \subseteq stack_{\preceq}\mathcal{IND}_A$ and for all $P_i, P_j \in \mathcal{P}$ it holds that $P_i \neq P_j$ for $i \neq j$,*

Since $stack_{\preceq}\mathcal{IND}_A$ provides the most general stage for inductive reasoning, we examine it in some detail now. First, observe that for an information system $\langle U, A, V, f\rangle$, it holds that:

$$stack_{\preceq}\mathcal{IND}_A = stack_{\preceq}\{IND(A)\}.$$

Thus, $\{IND(A)\}$ suffices to generate the whole $stack_{\preceq}\mathcal{IND}_A$. Furthermore, the stack operation allows us to connect $\mathcal{IND}_A$ with nearness type structures.

Definition 12. *Let X be a set and ν be a non-empty set of coverings of X such that:*

$$\mathcal{A} \preceq \mathcal{B} \,\&\, \mathcal{A} \in \nu \Rightarrow \mathcal{B} \in \nu.$$

Then (X, ν) is called a pre-nearness space.

When $stack_{\preceq}\mathcal{E}_\nu = \nu$, for

$$\mathcal{E}_\nu = \{\mathcal{P} \in \nu : \mathcal{P} \text{ is a partition of } X\},$$

then (X, ν) is called a *non-Archimedean pre-nearness space* and $\mathcal{E}_\nu$ is its *base*. Thus, the non-Archimedean structure $\mathcal{IND}_A$ on U is a base of the non-Archimedean pre-nearness space $(X, stack_{\preceq}\mathcal{IND}_A)$.

Definition 13. *Let (X, ν) be a pre-nearness space such that:*

$$\mathcal{A} \in \nu \,\&\, \mathcal{B} \in \nu \Rightarrow \{A \cap B : A \in \mathcal{A} \textit{ and } B \in \mathcal{B}\} \in \nu.$$

Then (X, ν) is called a merotopic space.

Definition 14. *A merotopic space (X, ν) which satisfies:*

$$\mathcal{A} \in \nu \Rightarrow \{Int_\nu(A) : A \in \mathcal{A}\} \in \nu,$$

where $Int_\nu(A) = \{x \in X : \{A, X \setminus \{x\}\} \in \nu\}$, is called a nearness space.

Proposition 3. *Let $\langle U, A, V, f\rangle$ be an information system. Then $(U, stack_{\preceq}\mathcal{IND}_A)$ is a non-Archimedean nearness space.*

Proof. In order not to overload the paper with definitions, we shall give just a hint how to prove this theorem. First, as is well-known, every partition star-refines itself. Therefore $(U, stack_{\preceq}\mathcal{IND}_A)$ is a uniform pre-nearness space. Second, every uniform pre-nearness space satisfies Definition 14, see, e.g. [4] for the proof. Finally, since $U/IND(A) = \mathcal{E}_{stack_{\preceq}\mathcal{IND}_A}$, the uniform pre-nearness space $(U, stack_{\preceq}\mathcal{IND}_A)$ is closed under intersections as required by Definition 13. Thus, $(U, stack_{\preceq}\mathcal{IND}_A)$ is a non-Archimedean nearness space.

Please, observe that the very simple description of SCM in terms of subsets of U have led us to $(U, stack_{\preceq}\mathcal{IND}_A)$ as a proper stage for human-like inductive reasoning. Surprisingly, this stage is nothing more than a representation of a basic concept of RST. Let us recall that any information system $\langle U, A, V, f\rangle$ may be regarded as a finite approximation space $(U, IND/A)$, and in many cases this representation is more handy, e.g. in algebraic investigations into RST. Actually, the same remark may be applied to $(U, stack_{\preceq}\mathcal{IND}_A)$.

Proposition 4. *Let U be a non-empty finite set. Then there is one-to-one correspondence between finite approximation spaces (U, E) and non-Archimedean nearness spaces (U, ν) over U.*

Proof. For the same reason as above, we also give a sketch of the proof. Every finite non-Archimedean nearness space (U, ν) is induced by a partition $\mathcal{P}$. Since $\mathcal{P}$ is the minimal open basis for the topology induced by Int_ν, it follows that (U, ν) is a topological nearness space. On the other hand, every finite topological space (U, ν) has the minimal open basis $\mathcal{P}$ for its topology Int_ν. Since is Int_ν symmetric, $\mathcal{P}$ is a partition and thus (U, ν) is a non-Archimedean nearness space. Finally, there is one-to-one correspondence between finite topological nearness spaces and finite approximation spaces. See also [19].

Thus, non-Archimedean nearness spaces over finite sets may be considered as another special representation of information systems. Approximation spaces are useful when one consider, e.g. relational structures and modal logics, whereas nearness spaces are suitable for ontologies and inductive reasoning.

6 Final Remarks

The article presents an account of preliminary results concerning Rough Set Theory (RST) and Similarity Coverage Model of category-based induction (SCM). In the first part of this paper we have shown how decision rules may be regarded as induction tasks and how rough inclusion functions may be used to compute the strength of inductive reasoning. In the second part we have presented a model of SCM based on non-Archimedean structures and non-Archimedean nearness spaces. Recently a number of attempts have been made to connect RST with nearness type structures, e.g. [11,19]. Thus, the paper has presented some intuitive reasons to consider these abstract topological spaces. The model based on a non-Archimedean space has a nice property that every ontology over it is taxonomic.

Acknowledgement. The research has been supported by the grant N N516 368334 from Ministry of Science and Higher Education of the Republic of Poland and by the grant Innovative Economy Operational Programme 2007-2013 (Priority Axis 1. Research and development of new technologies) managed by Ministry of Regional Development of the Republic of Poland.

References

1. Albatineh, A., Niewiadomska-Bugaj, M., Mihalko, D.: On Similarity Indices and Correction for Chance Agreement. Journal of Classification 23, 301–313 (2006)
2. Atran, S.: Classifying Nature Across Cultures. In: Osherson, D., Smith, E. (eds.) An Invitation to Cognitive Science. Thinking, pp. 131–174. MIT Press, Cambridge (1995)
3. Deses, D., Lowen-Colebunders, E.: On Completeness in a Non-Archimedean Setting via Firm Reflections. Bulletin of the Belgian Mathematical Society, Special volume: p-adic Numbers in Number Theory, Analytic Geometry and Functional Analysis, 49–61 (2002)
4. Deses, D.: Completeness and Zero-dimensionality Arising from the Duality Between Closures and Lattices, Ph.D. Thesis, Free University of Brussels (2003), `http://homepages.vub.ac.be/~diddesen/phdthesis.pdf`
5. Gomolińska, A.: On Three Closely Related Rough Inclusion Functions. In: Kryszkiewicz, M., Peters, J., Rybiński, H., Skowron, A. (eds.) RSEISP 2007. LNCS, vol. 4585, pp. 142–151. Springer, Heidelberg (2007)
6. Heit, E.: Properties of Inductive Reasoning. Psychonomic Bulletin & Review 7, 569–592 (2000)
7. Kloos, H., Sloutsky, V., Fisher, A.: Dissociation Between Categorization and Induction Early in Development: Evidence for Similarity-Based Induction. In: Proceedings of the XXVII Annual Conference of the Cognitive Science Society (2005)
8. Kulczyński, S.: Die Pflanzenassociationen der Pieninen. Bulletin International de L'Academie Polonaise des Sciences et des letters, classe des sciences mathematiques et naturelles, Serie B, Supplement II 2, 57–203 (1927)
9. Osherson, D.N., Smith, E.E., Wilkie, O., Lopez, A., Shafir, E.: Category-Based Induction. Psychological Review 97(2), 185–200 (1990)
10. Pawlak, Z.: Rough Sets. International Journal of Computer and Information Sciences 11, 341–356 (1982)
11. Peters, J., Skowron, A., Stepaniuk, J.: Nearness of Objects: Extension of Approximation Space Model. Fundamenta Informaticae 79, 497–512 (2007)
12. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning About Data. Kluwer Academic Publishers, Boston (1991)
13. Polkowski, L., Skowron, A.: Rough Mereology: A New Paradigm for Approximate Reasoning. Int. J. Approx. Reasoning 15(4), 333–365 (1996)
14. Skowron, A., Stepaniuk, J.: Tolerance Approximation Spaces. Fundamenta Informaticae 27, 245–253 (1996)
15. Sloman, S.A.: Feature-Based Induction. Cognitive Psychology 25, 231–280 (1993)
16. Tsumoto, S.: Extraction of Experts' Decision Rules from Clinical Databases Using Rough Set Model. Intelligent Data Analysis 2(1-4), 215–227 (1998)
17. Wille, R.: Restructuring Lattice Theory: an Approach Based on Hierarchies of Concepts. In: Rival, I. (ed.) Ordered Sets, pp. 445–470. Reidel, Dordrecht-Boston (1982)
18. Wille, R.: Concept lattices and Conceptual Knowledge Systems. Computers & Mathematics with Applications 23, 493–515 (1992)
19. Wolski, M.: Approximation Spaces and Nearness Type Structures. Fundamenta Informaticae 79, 567–577 (2007)

Probabilistic Dependencies in Linear Hierarchies of Decision Tables

Wojciech Ziarko

Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada S4S 0A2

Abstract. The article is a study of probabilistic dependencies between attribute-defined partitions of a universe in hierarchies of probabilistic decision tables. The dependencies are expressed through two measures: the probabilistic generalization of the Pawlak's measure of the dependency between attributes and the expected certainty gain measure introduced by the author. The expected certainty gain measure reflects the subtle grades of probabilistic dependence of events. Both dependency measures are developed and it is shown how they can be extended from flat decision tables to dependencies existing in hierarchical structures of decision tables.

1 Introduction

The notion of decision table has been around for long time and was widely used in circuit design, software engineering, business, and other application areas. In the original formulation, decision tables are static due to the lack of the ability to automatically learn and adapt their structures based on new information.

Decision tables representing data-acquired classification knowledge have been introduced by Pawlak [1]. In Pawlak's approach, the decision tables are dynamic structures derived from data, with the ability to adjust with new information. This fundamental difference makes it possible for novel uses of decision tables in applications related to reasoning from data, such as data mining, machine learning or complex pattern recognition.

The decision tables are typically used for making predictions about the value of the target decision attribute, such as medical diagnosis, based on combinations of values of condition attributes, for example symptoms and test results, as measured on new, previously unseen objects (for example, patients). However, the decision tables often suffer from the following problems related to the fact that they are typically computed based on a subset, a sample of the universe of all possible objects.

Firstly, the decision table may have excessive decision boundary, often due to poor quality of the descriptive condition attributes, which may be weakly correlated with the decision attribute. The excessive decision boundary leads to the excessive number of incorrect predictions.

Secondly, the decision table may be highly incomplete, i.e. excessively many new measurement vectors of condition attributes of new objects are not matched by any combination of condition attribute values present in the decision table. Such a highly incomplete decision table leads to an excessive number of new, unrepresented observations, for which the prediction of the decision attribute value is not possible.

With condition attributes weakly correlated with the decision attribute, increasing their number does not rectify the first problem. Attempting to increase the number of condition attributes, or the number of possible values of the attributes, results in the exponential explosion of the complexity of decision table learning and is leading to the rapid increase of its degree of incompleteness [8]. In general, the decision boundary reduction problem is conflicting with the decision table incompleteness minimization problem.

To deal with these fundamental difficulties, an approach involving building hierarchies of decision tables was proposed [6]. The approach is focused on learning hierarchical structures of decision tables rather than learning individual tables, subject to learning complexity constraints. In this approach, a linear hierarchy of decision tables is formed, in which the parent layer decision boundary defines a universe of discourse for the child layer table. The decision tables on each layer are size-limited by reducing the number of condition attributes and their values, thus bounding their learning complexity [8]. Each layer contributes a degree of decision boundary reduction, while providing a shrinking decision boundary to the next layer. In this way, even in the presence of relatively weak condition attributes, a significant total boundary reduction can be achieved, while preserving the learning complexity constraints on each level.

Similar to single layer decision table, the hierarchy of decision tables needs to be evaluated from the point of view of its quality as a potential classifier of new observations. The primary evaluative measure for decision tables, as introduced by Pawlak, is the measure of partial functional dependency between attributes [1] and its probabilistic extension [7]. Another measure is the recently introduced expected gain measure which captures more subtle probabilistic associations between attributes [7]. In this paper, these measures are reviewed and generalized to the hierarchical structures of decision tables. A simple recursive method of their computation is also discussed. The measures, referred to as γ and λ measures respectively, provide a tool for assessment of decision table-based classifiers derived from data.

The basics of the rough set theory and the techniques for analysis of decision tables are presented in this article in the probabilistic context, with the underlying assumption that the universe of discourse U is potentially infinite and is known only partially through a finite collection of observation vectors (the sample data). This assumption is consistent with great majority of applications in the areas of statistical analysis, data mining and machine learning.

2 Attribute-Based Probabilistic Approximation Spaces

In this section, we briefly review the essential assumptions, definitions and notations of the rough set theory in the context of probability theory.

2.1 Attributes and Classifications

We assume that observations about objects are expressed through values of *attributes*, which are assumed to be functions $a : U \to V_a$, where V_a is a finite set of values called the *domain.* The attributes represent some properties of the objects in U. It should be however mentioned here that, in practice, the attributes may not be functions but general relations due to influence of measurement random noise. The presence of noise may cause the appearance of multiple attribute values associated with an object.

Traditionally, the attributes are divided into two disjoint categories: *condition attributes* denoted as C, and *decision attributes* $D = \{d\}$. In many rough set-oriented applications, attributes are finite-valued functions obtained by discretizing values of real-valued variables representing measurements taken on objects $e \in U$.

As individual attributes, any non-empty subset of attributes $B \subseteq C \cup D$ defines a mapping from the set of objects U into the set of vectors of values of attributes in B. This leads to the idea of the equivalence relation on U, called indiscernibility relation $IND_B = \{(e_1, e_2) \in U : B(e_1) = B(e_2)\}$. According to this relation, objects having identical values of attributes in B are equivalent, that is, indistinguishable in terms of values of attributes in B . The collection of classes of identical objects will be denoted as U/B and the pair $(U, U/B)$ will be called an *approximation space.*

The object sets $G \in U/C \cup D$, will be referred to as *atoms.* The sets $E \in U/C$ will be referred to as *elementary sets.* The sets $X \in U/D$ will be called *decision categories.* Each elementary set $E \in U/C$ and each decision category $X \in U/D$ is a union of some atoms. That is, $E = \cup\{G \in U/C \cup D : G \subseteq E\}$ and $X = \cup\{G \in U/C \cup D : G \subseteq F\}$.

2.2 Probabilities

We assume that all subsets $X \subseteq U$ under consideration are measurable by a probability measure function P, normally estimated from collected data in a standard way, with $0 < P(X) < 1$, which means that they are likely to occur but their occurrence is not certain. In particular, each atom $G \in U/C \cup D$ is assigned a *joint probability* $P(G)$.

From our initial assumption and from the basic properties of the probability measure P, follows that for all atoms $G \in U/C \cup D$, we have $0 < P(G) < 1$ and $\sum_{G \in U/C \cup D} P(G) = 1$. Based on the joint probabilities of atoms, probabilities of elementary sets E and of a decision category X can be calculated by $P(E) = \sum_{G \subseteq E} P(G)$.

The probability $P(X)$ of the decision category X in the universe U is the *prior probability* of the category X. It represents the degree of confidence in the occurrence of the decision category X, in the absence of any information expressed by attribute values.

The *conditional probability* of a decision category X, $P(X|E) = \frac{P(X \cap E)}{P(E)}$, conditioned on the occurrence of the elementary set E, represents the degree of confidence in the occurrence of the decision category X, given information indicating that E occurred. The conditional probability can be expressed in terms of joint probabilities of atoms by $P(X|E) = \frac{\sum_{G \subseteq X \cap E} P(G)}{\sum_{G \subseteq E} P(G)}$. This property allows for simple computation of the conditional probabilities of decision categories.

2.3 Variable Precision Rough Sets

The theory of rough set underlies the methods for derivation, optimization and analysis of decision tables acquired from data. In this part, we review the basic definitions and assumptions of the variable precision rough set model (VPRSM) [5][7]. The VPRSM is a direct generalization of Pawlak rough sets [1]. One of the main objectives of rough set theory is the formation and analysis of approximate definitions of otherwise undefinable sets [1]. The approximate definitions, in the form of lower approximation and boundary area of a set, allow for determination of an object's membership in a set with varying degrees of certainty. The lower approximation permits for uncertainty-free membership determination, whereas the boundary defines an area of objects which are not certain, but possible, members of the set [1]. The VPRSM extends upon these ideas by parametrically defining the positive region as an area where the certainty degree of an object's membership in a set is relatively high, the negative region as an area where the certainty degree of an object's membership in a set is relatively low, and by defining the boundary as an area where the certainty of an object's membership in a set is deemed neither high nor low.

The defining criteria in the VPRSM are expressed in terms of conditional probabilities and of the prior probability $P(X)$ of the set X in the universe U. The prior probability $P(X)$ is used as reference value here as it represents the likelihood of X occurrence in the extreme case characterized by the absence of any attribute-based information. In the context the attribute-value representation of sets of the universe U, as described in the previous section, we will assume that the sets of interest are decision categories $X \in U/D$. Two *precision control* parameters are used: the *lower limit* l, $0 \leq l < P(X) < 1$, representing the highest acceptable degree of the conditional probability $P(X|E)$ to include the elementary set E in the *negative region* of the set X; and the *upper limit* u, $0 < P(X) < u \leq 1$, reflecting the least acceptable degree of the conditional probability $P(X|E)$ to include the elementary set E in the positive region, or *u-lower approximation* of the set X. The *l-negative region* of the set X, denoted as $NEG_l(X)$ is defined by:

$$NEG_l(X) = \cup\{E : P(X|E) \leq l\} \tag{1}$$

The l-negative region of the set $X$is a collection of objects for which the probability of membership in the set X is *significantly lower* than the prior probability $P(X)$. The u-positive region of the set X, $POS_u(X)$ is defined as

$$POS_u(X) = \cup\{E : P(X|E) \geq u\}. \tag{2}$$

The u-positive region of the set X is a collection of objects for which the probability of membership in the set X is *significantly higher* than the prior probability

$P(X)$. The objects which are not classified as being in the u-positive region nor in the l-negative region belong to the (l,u)-boundary region of the decision category X, denoted as

$$BNR_{l,u}(X) = \cup\{E : l < P(X|E) < u\}. \tag{3}$$

The boundary is a specification of objects about which it is known that their associated probability of belonging, or not belonging to the decision category X, is not much different from the prior probability of the decision category $P(X)$. The VPRSM reduces to standard rough sets when $l = 0$ and $u = 1$.

3 Structures of Decision Tables Acquired from Data

To describe functional or partial functional connections between attributes of objects of the universe U, Pawlak introduced the idea of decision table acquired from data [1]. The probabilistic decision tables and their hierarchies extend this idea into probabilistic domain by forming representations of probabilistic relations between attributes.

3.1 Probabilistic Decision Tables

For the given decision category $X \in U/D$ and the set values of the VPRSM lower and upper limit parameters l and u, we define the *probabilistic decision table* $DT^{C,D}_{l,u}$ as a mapping $C(U) \rightarrow \{POS, NEG, BND\}$ derived from the classification table as follows:

The mapping is assigning each tuple of values of condition attribute values $t \in C(U)$ to its unique designation of one of VPRSM approximation regions $POS_u(X)$, $NEG_l(X)$ or $BND_{l,u}(X)$, the corresponding elementary set E_t is included in, along with associated elementary set probabilities $P(E_t)$ and conditional probabilities $P(X|E_t)$:

$$DT^{C,D}_{l,u}(t) = \begin{cases} (P(E_t), P(X|E_t), POS) \Leftrightarrow E_t \subseteq POS_u(X) \\ (P(E_t), P(X|E_t), NEG) \Leftrightarrow E_t \subseteq NEG_l(X) \\ (P(E_t), P(X|E_t), BND) \Leftrightarrow E_t \subseteq BND_{l,u}(X) \end{cases} \tag{4}$$

The probabilistic decision table is an approximate representation of the probabilistic relation between condition and decision attributes via a collection of uniform size probabilistic rules corresponding to rows of the table. An example probabilistic decision table is shown in Table 1. In this table, the condition attributes are *a, b, c*, attribute-value combinations correspond to elementary sets E and ***Region*** is a designation of one of the *approximation regions* the corresponding elementary sets belong to: positive (**POS**), negative (**NEG**) or boundary (**BND**).

The probabilistic decision tables are most useful for decision making or prediction when the relation between condition and decision attributes is largely non-deterministic. However, they suffer from the inherent contradiction between

Table 1. An example of probabilistic decision table

a	b	c	$P(E)$	$P(X\|E)$	***Region***
1	1	2	0.23	1.00	**POS**
1	0	1	0.33	0.61	**BND**
2	2	1	0.11	0.27	**BND**
2	0	2	0.01	1.00	**POS**
0	2	1	0.32	0.06	**NEG**

the accuracy and completeness. In the presence of boundary region, higher accuracy, i.e. reduction of boundary region, can be achieved either by adding new condition attributes or by increasing the precision of existing ones (for instance, by making the discretization procedure finer). Both solutions lead to the exponential growth in the maximum number of attribute-value combinations to be stored in the decision table [8]. In practice, it results in such negative effects as excessive size of the decision table, likely high degree of table incompleteness (in the sense of missing many feasible attribute-value combinations), weak data support for elementary sets represented in the table and, consequently, unreliable estimates of probabilities. The use of hierarchies of decision tables rather than individual tables in the process of classifier learning from data provides a partial solution to these problems [6].

3.2 Probabilistic Decision Table Hierarchies

Since the VPRSM boundary region $BND_{l,u}(X)$ is a definable subset of the universe U, it allows to structure the decision tables into hierarchies by treating the boundary region $BND_{l,u}(X)$ as sub-universe of U, denoted as $U' = BND_{l,u}(X)$. The "child" sub-universe U' so defined can be made completely independent from its "parent" universe U, by having its own collection of condition attributes C' to form a "child" approximation sub-space $(U, U/C')$. As on the parent level, in the approximation space $(U, U/C')$, the decision table for the subset $X' \subseteq X$ of the target decision category X, $X' = X \cap BND_{l,u}(X)$ can be derived by adapting the formula (4). By repeating this step recursively, a linear hierarchy of probabilistic decision tables can be grown until either boundary area disappears in one of the child tables, or no attributes can be identified to produce non-boundary decision table at the final level. Other termination conditions are possible, but this issue is out of scope in this article.

The nesting of approximation spaces obtained as a result of recursive computation of decision tables, as described above, creates a new approximation space on U. The resulting *hierarchical approximation space* (U, R) cannot be expressed by the indiscernibility relation, as defined in Section 2, in terms of the attributes used to form the local sub-spaces on individual levels of the hierarchy. This leads to the basic question: how to measure the degree of the mostly probabilistic dependency between the *hierarchical partition* R of U and the partition $(X, \neg X)$ corresponding to the decision category $X \subseteq U$. Some probabilistic inter-partition dependency measures are explored in the next section.

4 Dependencies in Decision Table Hierarchies

The dependencies between partitions are fundamental to rough set-based non-probabilistic and probabilistic reasoning and prediction. They allow to predict the occurrence of a class of one partition based on the information that a class of another partition occurred. There are several ways dependencies between partitions can be defined in decision tables. In Pawlak's early works functional and partial functional dependencies were explored [1]. The probabilistic generalization of the dependencies was also defined and investigated in the framework of the variable precision rough set model. All these dependencies represent the relative size of the positive and negative regions of the target set X. They reflect the quality of approximation of the target category in terms of the elementary sets of the approximation space. Following the original Pawlak's terminology, we will refer to these dependencies as γ-*dependencies*.

Other kind of dependencies, based on the notion of the certainty gain measure, reflect the average degree of improvement of the certainty of occurrence of the decision category X, or $\neg X$, relative to its prior probability $P(X)$ [7] (see also [2] and [4]). We will refer to these dependencies as λ-*dependencies*. Both, the γ-dependencies and λ-dependencies can be extended to hierarchies of probabilistic decision tables, as described below. Because there is no single collection of attributes defining the partition of U, the dependencies of interest in this case are dependencies between the *hierarchical partition* R generated by the decision table hierarchy, forming the approximation space (U, R), and the partition $(X, \neg X)$, defined by the target set.

4.1 Γ-Dependencies for Decision Tables

The partial functional dependency among attributes, referred to as γ-*dependency* $\gamma(D|C)$ measure, was introduced by Pawlak [1]. It can be expressed in terms of the probability of positive region of the partition U/D defining decision categories:

$$\gamma(D|C) = P(POS^{C,D}(U)) \tag{5}$$

where $POS^{C,D}(U)$ is a positive region of the partition U/D in the approximation space induced by the partition U/C. In the binary case of two decision categories, X and $\neg X$, the $\gamma(D|C)$-dependency can be extended to the VPRSM by defining it as the combined probability of the u-positive and l-negative regions:

$$\gamma_{l,u}(X|C) = P(POS_u(X) \cup NEG_l(X)). \tag{6}$$

The γ-dependency measure reflects the proportion of objects in U, which can be classified with sufficiently high certainty as being members, or non-members of the set X.

4.2 Computation of Γ-Dependencies in Decision Table Hierarchies

In the case of the approximation space obtained by forming it via hierarchical classification process, the γ-dependency between the hierarchical partition R and the partition $(X, \neg X)$ can be computed directly by analyzing all classes of the hierarchical partition. However, an easier to implement recursive computation is also possible. This is done by recursively applying, starting from the leaf table of the hierarchy and going up to the root table, the following formula (7) for computing the dependency of the parent table $\gamma^U_{l,u}(X|R)$ in the hierarchical approximation space (U, R), if the dependency of a child level table $\gamma^{U'}_{l,u}(X|R')$ in the sub-approximation space (U', R') is given:

$$\gamma^U_{l,u}(X|R) = \gamma^U_{l,u}(X|C) + P(U')\gamma^{U'}_{l,u}(X|R'), \tag{7}$$

where C is a collection of attributes inducing the approximation space U and $U' = BND_{l,u}(X)$. As in the flat table case, this dependency measure represents the fraction of objects that can be classified with acceptable certainty into decision categories X or $\neg X$ by applying the decision tables in the hierarchy. The dependency of the whole structure of decision tables, that is the last dependency computed by the recursive application of formula (7), will be called a *global* γ-*dependency*. Alternatively, the global γ-dependency can be computed straight from from the definition (5). This computation requires checking all elementary sets of the hierarchical partition for the inclusion in $POS_u(X) \cup NEG_l(X)$, which seems to be less elegant and more time consuming that the recursive method.

4.3 Certainty Gain Functions

Based on the probabilistic information contained in data, as given by the joint probabilities of atoms, it is also possible to evaluate the degree of probabilistic dependency between any elementary set and a decision category. The dependency measure is called *absolute certainty gain* [7] (*gabs*). It represents the degree of influence the occurrence of an elementary set E has on the likelihood of occurrence of the decision category X. The occurrence of E can increase, decrease, or have no effect on the probability of occurrence of X. The probability of occurrence of X, in the absence of any other information, is given by its prior probability $P(X)$. The degree of variation of the probability of X, due to occurrence of E, is reflected by *the absolute certainty gain function:*

$$gabs(X|E) = |P(X|E) - P(X)|, \tag{8}$$

where $| * |$ denotes absolute value function. The values of the absolute gain function fall in the range $0 \leq gabs(X|E) \leq max(P(\neg X), P(X)) < 1$. In addition, if sets X and E are independent in the probabilistic sense, that is, if $P(X \cap E) = P(X)P(E)$, then $gabs(X|E) = 0$. The definition of the absolute certainty gain provides a basis for the definition of a new probabilistic dependency measure between attributes. This dependency can be expressed as the average degree of change of occurrence certainty of the decision category X, or of its complement $\neg X$, due to occurrence of any elementary set [7], as defined by the *expected certainty gain* function:

$$egabs(X|C) = \sum_{E \in U/C} P(E)gabs(X|E), \tag{9}$$

where $X \in U/D$. The expected certainty gain is a more subtle inter-partition dependency than γ-dependency since it takes into account the probabilistic distribution information in the boundary region of X. The $egabs(X|C)$ measure can be computed directly from joint probabilities of atoms. It can be proven [7] that the expected gain function falls in the range $0 \leq egabs(X|C) \leq 2P(X)(1 - P(X))$, where $X \in U/D$.

4.4 Attribute Λ-Dependencies in Decision Tables

The strongest dependency between attributes of a decision table occurs when the decision category X is definable, i.e. when the dependency is functional. Consequently, the dependency in this deterministic case can be used as a reference value to normalize the certainty gain function. The following normalized expected gain function $\lambda(X|C)$ measures the expected degree of the probabilistic dependency between elementary sets and the decision categories belonging to U/D [7]:

$$\lambda(X|C) = \frac{egabs(X|C)}{2P(X)(1 - P(X))}, \tag{10}$$

where $X \in U/D$. The λ-dependency quantifies in relative terms the average degree of deviation of elementary sets from statistical independence with the decision class $X \in U/D$. The dependency function reaches its maximum $\lambda(X|C) = 1$ only if the dependency is deterministic (functional) and is at minimum when all events represented by elementary sets $E \in U/C$ are unrelated to the occurrence of the decision class $X \in U/D$. In the latter case, the conditional distribution of the decision class $P(X|E)$ equals to its prior distribution $P(X)$.

The value of the $\lambda(X|C)$ dependency function can be easily computed from the joint probabilities of atoms. As opposed to the generalized $\gamma(X|C)$ dependency, the $\lambda(X|C)$ dependency has the *monotonicity property* [3], that is, $\lambda(X|C) \leq \lambda(X|C \cup \{a\})$, where a is an extra condition attribute outside the set C. This monotonicity property allows for dependency-preserving reduction of attributes and is leading to the notion of probabilistic λ-reduct of attributes, as defined in [3].

4.5 Computation of Λ-Dependencies in Decision Table Hierarchies

The λ-dependencies can be computed directly based on any known partitioning of the universe U. In cases when the approximation space is formed through hierarchical classification, the λ-dependency between the partition R so created and the target category X can be computed via a recursive formula derived below. Let

$$egabs_{l,u}(X|C) = \sum_{E \in POS_u \cup NEG_l} P(E) gabs(X|E) \tag{11}$$

denote the conditional expected gain function, i.e. restricted to the union of positive and negative regions of the target set X in the approximations space generated by attributes C. The maximum value of $egabs_{l,u}(X|C)$, achievable in deterministic case, is $2P(X)(1 - P(X))$. Thus, the normalized *conditional λ-dependency* function, can be defined as:

$$\lambda_{l,u}(X|C) = \frac{egabs_{l,u}(X|C)}{2P(X)(1 - P(X))}. \tag{12}$$

As γ-dependencies, λ-dependencies between the target partition $(X, \neg X)$ and the hierarchical partition R can be computed recursively. The following formula (13) describes the relationship between λ-dependency computed in the approximation space (U, R), versus the dependency computed over the approximation sub-space (U, R'), where R and R' are hierarchical partitions of universes U and $U' = BND_{l,u}(X)$, respectively. Let $\lambda_{l,u}(X|R)$ and $\lambda_{l,u}(X|R')$ denote λ-dependency measures in the approximation spaces (U, R) and (U', R'), respectively. The λ-dependencies in those approximation spaces are related by the following:

$$\lambda_{l,u}(X|R) = \lambda_{l,u}(X|C) + P(BND_{l,u}(X))\lambda_{l,u}(X|R'). \tag{13}$$

The proof of the above formula follows directly from the Bayes's equation. In practical terms, the formula (13) provides a method for efficient computation of conditional λ-dependency in a hierarchical arrangement of probabilistic decision tables. According to this method, to compute conditional λ-dependency for each level of the hierarchy, it suffices to compute the conditional λ-dependency and to know "child" $BND_{l,u}(X)$-level conditional λ-dependency. That is, the conditional λ-dependency should be computed first for the bottom level table using formula (12), and then it would be computed for each subsequent level in the bottom-up fashion by successively applying (13).

In similar way, the "unconditional" λ-dependency $\lambda(X|R)$ can be computed over all elementary sets of the hierarchical approximation space. This is made possible by the following variant of the formula (13):

$$\lambda(X|R) = \lambda_{l,u}(X|C) + P(BND_{l,u}(X))\lambda(X|R'). \tag{14}$$

The recursive process based on the formula (14) is essentially the same as in the case (13), with except that the bottom-up procedure starts with computation of the "unconditional" λ-dependency by formula (10) for the the bottom-level table.

5 Concluding Remarks

Learning and evaluation of hierarchical structures of probabilistic decision tables is the main focus of this article. The earlier introduced measures of gamma and lambda dependencies between attributes [7] for decision tables acquired from data are not directly applicable to approximation spaces corresponding to hierarchical structures of decision tables. The main contribution of this work is the extension of the measures to the decision table hierarchies case and the derivation of recursive formulas for their easy computation. The gamma dependency measure allows for the assessment of the prospective ability of the classifier based on the hierarchy of decision tables to predict the values of decision attribute on required level of certainty. The lambda dependency measure captures the relative degree of probabilistic correlation between classes of the partitions corresponding to condition and decision attributes, respectively. The degree of the correlation in this case is a representation of the average improvement of the ability to predict the occurrence of the target set X, or its complement $\neg X$. Jointly, both measures enable the user to evaluate the progress of learning with the addition of new training data and to assess the quality of the empirical classifier. Three experimental applications of the presented approach are currently under development. The first one is concerned with face recognition using photos to develop the classifier in the form of a hierarchies of decision tables, the second one is aiming at adaptive learning of spam recognition among e-mails, and the third one is focused on stock price movement prediction using historical data.

Acknowledgment. This paper is an extended version of the article included in the Proceedings of the International Conference on Rough Sets and Emerging Intelligent Systems Paradigms, devoted to the memory of Professor Zdzislaw Pawlak, held in Warsaw, Poland in 2007. The support of the Natural Sciences and Engineering Research Council of Canada in funding the research presented in this article is gratefully acknowledged.

References

1. Pawlak, Z.: Rough sets - Theoretical Aspects of Reasoning About Data. Kluwer, Dordrecht (1991)
2. Greco, S., Matarazzo, B., Slowinski, R.: Rough membership and Bayesian confirmation measures for parametrized rough sets. In: Ślęzak, D., Wang, G., Szczuka, M.S., Düntsch, I., Yao, Y. (eds.) RSFDGrC 2005. LNCS, vol. 3641, pp. 314–324. Springer, Heidelberg (2005)
3. Slezak, D., Ziarko, W.: The Investigation of the Bayesian rough set model. International Journal of Approximate Reasoning 40, 81–91 (2005)
4. Yao, Y.: Probabilistic approaches to rough sets. Expert Systems 20(5), 287–291 (2003)
5. Ziarko, W.: Variable precision rough sets model. Journal of Computer and Systems Sciences 46(1), 39–59 (1993)
6. Ziarko, W.: Acquisition of hierarchy-structured probabilistic decision tables and rules from data. In: Proc. of IEEE Intl. Conf. on Fuzzy Systems, Honolulu, pp. 779–784 (2002)
7. Ziarko, W.: Probabilistic rough sets. In: Ślęzak, D., Wang, G., Szczuka, M.S., Düntsch, I., Yao, Y. (eds.) RSFDGrC 2005. LNCS, vol. 3641, pp. 283–293. Springer, Heidelberg (2005)
8. Ziarko, W.: On learnability of decision tables. In: Tsumoto, S., Słowiński, R., Komorowski, J., Grzymała-Busse, J.W. (eds.) RSCTC 2004. LNCS, vol. 3066, pp. 394–401. Springer, Heidelberg (2004)

Automatic Singing Voice Recognition Employing Neural Networks and Rough Sets

Paweł Żwan, Piotr Szczuko, Bożena Kostek, and Andrzej Czyżewski

Gdańsk University of Technology, Multimedia Systems Department
Narutowicza 11/12, 80-952 Gdańsk, Poland
{zwan,szczuko,bozenka,ac}@sound.eti.pg.gda.pl

Abstract. The aim of the research study presented in this paper is the automatic recognition of a singing voice. For this purpose, a database containing sample recordings of trained and untrained singers was constructed. Based on these recordings, certain voice parameters were extracted. Two recognition categories were defined – one reflecting the skills of a singer (quality), and the other reflecting the type of the singing voice (type). The paper also presents the parameters designed especially for the analysis of a singing voice and gives their physical interpretation. Decision systems based on artificial neutral networks and rough sets are used for automatic voice quality/ type classification. Results obtained from both decision systems are then compared and conclusions are derived.

Keywords: Singing Voice, Feature extraction, Automatic Classification, Artificial Neural Networks, Rough Sets, Music Information Retrieval.

1 Introduction

The area of automatic content indexing and classification is related to the Music Information Retrieval (MIR) domain, which is now growing very rapidly and induces many discussions on automatic speech recognition and the development of appropriate systems. The speech is not the only outcome of the human voice organ. Singing is another one, and is considered a musical instrument by musicologists. However, its artistic and musical aspects are the reason why singing must be analyzed by specially designed additional parameters. These parameters obviously should be based on speech parameters, but additionally they must focus on the articulation and the timbre.

A parametric description is necessary in many applications of automatic sound recognition. A very complicated biomechanics of the singing voice [10], [27] and a different character of the intonation and the timbre of the voice require numerous features to describe its operation. Such a parametric representation needs intelligent decision systems in the classification process. In the presented study, artificial neural network (ANN) and rough set-based (RS) decision systems were employed for the purpose of the singing voice quality/type recognition. The systems were trained with sound samples, of which a large part (1700 samples) was recorded in the studio and 1200 samples were extracted from professional CD recordings. For every sound sample, a feature vector (FV) containing 331 parameters was formed. The parameters were divided into two groups: the so-called dedicated ones (designed allowing for a singing voice specifics) and more general ones known from the literature on MIR and speech recognition. The decision system ability to automatically classify a singing voice is discussed in the context of comparing the efficiency of ANN and RS systems in two recognition categories: 'voice type' (classes: bas, baritone, tenor, alto, mezzo-soprano, soprano) and 'voice quality' (classes: amateur, semi-professional, professional). Additionally, the parameters were judged using statistical and rough set methods. For different methods of reducing the feature vector redundancy, new classifiers were trained. The results were compared by analyzing the accuracy of the trained recognition systems. This article is an extended version of the paper presented at the RSEISP'07 conference held in Warsaw [34].

The paper is organized as follows. In Section 2 the organization of the database of samples of singing is described. The automatic process of classification requires an efficient block of feature extraction, thus Section 3 presents parameters that were used in experiments and discusses them in the context of their relationship with voice production mechanisms. The analysis shown in Section 4 concentrates around the redundancy elimination in the feature vector. For this purpose three methods, i.e. Fisher and Sebestyen statistics, and the rough set-based method are employed. The main core of experiments is presented in Section 5, and finally Section 6 summarizes the results obtained in this study.

2 The Database of Singing Voices

The prepared singing voice database contains over 2900 sound samples. Some 1700 of them were recorded from 42 singers in a studio. The vocalists consisted of three groups: amateurs (Gdańsk University of Technology Choir vocalists), semi-professionals (Gdańsk Academy of Music, Vocal Faculty students), and professionals (qualified vocalists, graduated from the Vocal Faculty of the Gdańsk Academy of Music). Each of them recorded 5 vowels: 'a', 'e', 'i', 'o', 'u' at several sound pitches belonging to a natural voice scale. These recordings formed the first singing category – singing quality. The singing voice type category was formed by assigning the voices to one of the following classes: bas, baritone, tenor, alto, mezzo-soprano and soprano.

The second group of samples was prepared on the basis of CD audio recordings of famous singers. The database of professionals needed to be extended due to the fact that voice type recognition is possible only among professional voices. Amateur voices do not show many differences within groups of male and female voices as it has already been researched in literature [2], [27].

3 Parametrization of the Singing Voice

In order to parameterize the singing voice properly, the understanding of the voice production mechanism is required. The biomechanism of the singing voice creation is rather complicated, but in the domain of its spectral energy (while not taking phase changes into account) it can be simplified by assuming the FIR model of the singing voice production. As in each classical FIR model a vocal tract is a filter which changes the spectrum of a source (the glottal) by a sum of resonances with given frequencies, amplitudes and qualities (the vocal tract). Therefore the singing is produced by the vibration of human vocal cords and resonances in the throat and head cavities. The resonances produce formants in the spectrum of sounds. Formants are not only related to articulation, and enable to produce different vowels, but also characterize timbre and voice type qualities. For example the formant of the middle frequency band (3.5 kHz) is described in literature as "singer's formant", and its relation to voice quality is proved [2], [20], [27]. This concept is well recognized in a reach literature related to singing.

However, the interaction between two factors, namely glottal source and resonance characteristics, shapes the timbre and power of an outgoing vocal sound, and both factors are equally important. Thus, singing voice parameters can be divided into two groups related to those two factors. Since this is a classical FIR model in order to deconvolve from the output signal the source from the filter inverse filtration methods are required. In literature, some inverse filtration methods for deriving glottis parameters are presented, however they are proved to be inefficient due to phase problems [10]. In this aspect only parameters of vocal tract formants can be calculated directly from the inverse filtering analysis since they are defined in frequency. The assumption of linearity is the reason why time parameters of the source signal must be extracted by other methods which will be presented later on.

Vocal tract parameters are in a speech analysis most often derived from the LPC method, but an adequate separation of frequency resonances demands high resolution for lower frequencies, where the resonances are located. Moreover, the methods of analysis with a spectrum resolution controlled by a function of sound pitch are required. The warped LPC method [8], [18] (further called the WLPC analysis) fulfills those conditions and enables to analyze frequencies and levels of formants with a controlled higher low frequency resolution (below 5 kHz). It is based on nonlinear sampling of the unit circle in a z transform, thus the resolution in lower frequencies is better comparing to a classical LPC analysis with the same length of the analyzed frame. The phase response frequency is transformed non-linearly to a warped frequency ω_W according to Equation (1).

$$\omega_W = \omega + 2 \cdot \arctan\left(\frac{\lambda \cdot \sin\omega}{1 - \lambda \cdot \cos\omega}\right) \tag{1}$$

where λ is a parameter, which determines non-linearity of the transformation and low frequency resolution of the WLPC analysis. The larger λ is, the more densely are the lower frequencies sampled. Mathematical aspects of this transformation are presented in detail in some literature sources [8], [9], [18] and in the previous works of the authors of this paper [34].

Since the analysis is applied to small signal frames it can be performed for several parts of the analyzed sounds. Therefore, any parameter F (which can be for example the level of one of the formants) forms a vector which describes its values in consecutive frames. In order to focus on a whole signal, and not only on a single frame, the median value of this vector is represented by a so-called static parameter F_{med}. In this case, a median value is better then a medium value, because it is more resistant to short non typical values of a parameter, which do not drastically change the median value. On the other hand, in order to investigate the stability, the variances of the vector values (denoted as F_{var}) are also taken into account.

Some of the singing voice parameters must be calculated for a whole sound rather than for single frames. Those parameters are defined on the basis of the fundamental frequency contour analysis, and they are related to vibrato and intonation. Vibrato is described as the modulation of the fundamental frequency of sounds performed by singers in order to change timbre of sounds, while intonation is their ability to produce sounds perceived as stable and precise in tune.

The parameters based on the singing voice analysis ('dedicated' parameters) form an important group, but they should be supplemented with general descriptors normally used for the classification of sound instruments. This group of parameters was investigated in detail in the domain of automatic sound recognition at Multimedia Systems Department at Gdańsk University of Technology. The usefulness of those parameters in automatic musical sound recognition was proved, and implied their application to the field of the singing voice recognition.

In this study, 331 parameters were derived from the analyses, of which 124 are defined by the authors and are so-called 'dedicated parameters' especially designed to address signing voice specifics.

3.1 Estimation of the Vocal Tract Parameters

As already described, the estimation of formants requires methods of analysis with good frequency resolution which are dependent on pitch of sounds. If the resolution is not properly set, single harmonics can be erroneously recognized as formants. For those purposes the WLPC analysis seems to be the most appropriate because the λ parameter is the function of the pitch of analyzed sounds [9], and thus can be changed in this analysis. The function λ=f(f) is presented in Eq. (2). The problem of how to determine the appropriate λ is presented in detail in the work of one of the authors [32], [34].

$$\lambda = 10^{-6} \cdot f\,[Hz]^2 - 0.0022 \cdot f\,[Hz] + 0.9713 \qquad (2)$$

However, parameters related to the 'singing formant' can also be extracted on the basis of the FFT power spectrum parametrization. Correlation between the WLPC and FFT parameters is not a problematic issue. Various methods, among them statistical analysis and rough set method, enable to reduce redundancy in feature vectors (FVs) and to compare the significance of the features. The WLPC and FFT analyses results are presented in Fig. 1. Maxima and minima of the WLPC curves are determined automatically by an algorithm elaborated by one of the authors [32]. WLPC smoothes the power spectrum with a good resolution for frequencies below 5kHz.

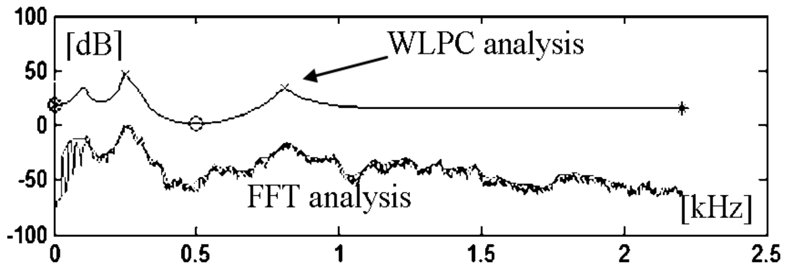


Fig. 1. WLPC analysis shown along with the FFT power spectrum analysis of sound

Extracted WLPC maxima are related to one of the three formants: articulation (frequencies 1-2.5kHz), singer's (singing) (frequencies 3-4 kHz), and high singing formants (frequencies over 5kHz). Since in literature a formal prescription of how to define mathematically these formants does not exist, three definitions for each of them can be proposed basing on three WLPC minima.

$$F_{nm} = WLPCmx_n - WLPCmn_m \tag{3}$$

where $WLPCmx_n$ is the value of the nth WLPC maximum and $WLPCmn_m$ is a value of the mth WLPC minimum.

Since the WLPC analysis is applied to short signal frames, it can be performed for several fragments of the analyzed sounds. Therefore, any formant parameter F_{nm} forms a vector which describes its values in consecutive frames. Median values of this vector represent a so-called static parameter F_{nmmed}, while the values of variances are a dynamic representation and are denoted as F_{nmvar}. The maximum and minimum values from expression (5) are also calculated respectively in all consecutive frames. They are denoted as F_{nmmax} and F_{nmmin} respectively.

A singing formant is presented by many authors as significant for the estimation of singing quality. Parameters related to the "singer's formant" were extracted on the basis of the linear combination of parameters F_{nm} and additionally by using the FFT power spectrum parametrization. The combinations of the parameters defined basing on the WLPC analysis are presented in Eq. (4) and (5). Those equations show direct relationship between formants. The parameter related to the formant energy defined on the basis of the FFT power spectrum is presented in (6).

$$\frac{F_2}{F_1} = F_{21} - F_{11} \tag{4}$$

$$\frac{F_2}{F_3} = F_{21} - F_{31} \tag{5}$$

$$SFL = \frac{E_{SF}}{E_{total}} \tag{6}$$

where ratios F_2/F_1 and F_2/F_3 represent a difference in formant levels F_{11}, F_{21} and F_{31} expressed in [dB], SFL denotes the singer's formant energy, E_{SF} is the power spectrum energy for the band (2.5kHz-4kHz) in which a 'singing formant' is present, and E_{total} is the total energy of the analyzed signal.

3.2 Estimation of the Glottal Source Parameters

Interaction between the vocal tract filter and the glottal shape along with phase problems are obstacles for an accurate automatic glottal source shape extraction [10], [12], [27], [32]. Glottal source parameters, which are defined in the time domain, are not easy to compute from the inverse filtration. However, within the context of the singing voice quality their stability rather that their objective values seems to be important. The analysis must be done for single periods of sound, and the sonogram analysis with small analyzing frames and big overlap should be employed.

For each of the frequency bands, the sonogram consists of a set of n sequences $S_n(k)$, where n is the number of a frequency band and k is the number of a sample. Since the aim of the parametrization is to describe the stability of energy changes in sub-bands, the autocorrelation in time is a function of sequences $S_n(k)$. The more frequent and stable energy changes in a sub-band were, the higher were the values of the autocorrelation function maximum (for index not equal to 0). The analysis was performed for 16 and 32 sample frames. In the first case the energy band of 0-10 kHz was related to the first four indexes n and the maximum of the autocorrelation function of sub-band n is denoted as KX_n (7), in the second case n=1...8 and the resulting parameter is defined as LX_n (8). Two different analyzing frames were used for comparison purposes only. The redundancy in the feature vector (FV) was further eliminated by statistical methods.

$$KX_n = \max(\underset{k}{Corr}(S_n^{16}(k))), \quad n = 1...4 \tag{7}$$

$$LX_n = \max(\underset{k}{Corr}(S_n^{32}(k))), \quad n = 1...8 \tag{8}$$

where $\underset{k}{Corr}(.)$ is the autocorrelation function in time domain, k – sample number, n - number of the frequency sub-band, S_n^{16} – sonogram samples sequence for the analyzed frame of 16 samples and frequency sub-band n, and S_n^{32} denotes a sonogram sample sequence for the analyzed frame of 32 samples and the frequency sub-band n.

Conversely, the minimum of the correlation $Corr(S_n(k))$ function is connected with the symmetry or anti-symmetry of energy changes in sub-bands, which relates to the open quotient of glottis source [32]. Therefore in each of the analyzed sub-bands the KY_n and LY_n parameters are defined as (9) and (10), respectively:

$$KY_n = \min(\underset{k}{Corr}(S_n^{16}(k))), \quad n = 1...4 \tag{9}$$

$$LY_n = \min(\underset{k}{Corr}(S_n^{32}(k))), \quad n = 1...8 \tag{10}$$

where $\underset{k}{Corr}(.)$, k, n, S_n^{16}, S_n^{32} are defined as in formulas (7) and (8).

Another parameter defined for each analyzed sub-band is a threshold parameter KP_n defined as the number of samples exceeding the average energy level of the sub-band n divided by the total number of samples in the sub-band. For the frame of 32 samples a similar parameter is defined and denoted as LP_n. Parameters KP_n and LP_n are also related to the open quotient of the glottal signal [32], [33].

3.3 Estimation of Intonation and Vibrato Parameters

Proper vibrato and intonation play a very important role in the perception of voice quality [4], [5], [7], [25], [31]. It is clear that a person who does not hold the pitch steadily and does not have a consistent vibrato cannot be judged as a good singer. Intonation and vibrato of the singing is defined in the frequency domain, thus a pitch contour needs to be extracted.

There are several methods of automatic sound pitch extraction, of which autocorrelation method seems to be the most appropriate [6], [14]. Autocorrelation pitch extraction method is based on the determination of the maximum of an autocorrelation function defined for the overlapped segments of the audio signal. Since this method is well presented in a reach literature [6], [23] on this subject, it will not be recalled here. The fundamental frequency (f_0) within each analyzed frame was determined, and at the same time the improvement of the frequency resolution of the analysis was achieved by interpolating three samples around the maximum of the autocorrelation function. The length of the frame was set to 512 samples. The value was determined experimentally in order to give a satisfactory time resolution. It is presented in detail in other papers of the authors [6], [32]. The interpolation improves the frequency resolution significantly.

The pitch of the analyzed sounds is not always stable in time, especially when sounds of untrained singers are concerned. In order to accurately parametrize vibrato and intonation of the analyzed sound, an equivalent pitch contour of the sound but without vibrato should be determined. The result of such analysis is a so-called 'base contour' which is calculated by smoothing the pitch contour (using the moving average method) with the frame length equal to the reciprocal of the half vibrato frequency. When $bc(n)$ are samples of the base contour (defined in frequency) and $v(n)$ are samples of the vibrato contour, the vibrato modified contour is calculated as $v_m(n) = v(n)\text{-}bc(n)$ and it is used for the vibrato parametrization. On the other hand, $bc(n)$ are used for the intonation parametrization to define how quickly the singer is able to obtain a given pitch of the sound and how stable its frequency is.

The parametrization of vibrato depth and frequency (f_{VIB}) may be not sufficient in the category of singing quality. Since the stability of vibrato reflects the quality of sound parameters in time [5], [27], additional three vibrato parameters were defined [5], [34]:

– *"perdiodicity"* of vibrato VIB_P (Eq. 11) pitch contour, defined as the maximum value of the autocorrelation of the pitch contour function (for index not equal to 0);

- *"harmonicity"* of vibrato VIB_H (Eq. 12) obtained by calculating *Spectrum Flatness Measure* for the spectrum of the pitch contour;
- *"sinusoidality"* of vibrato VIB_S (Eq. 13) defined as the similarity of the parameterized pitch contour to the sine waveform.

$$VIB_P = \max_n (Corr(f_0(n)) \tag{11}$$

$$VIB_H = \frac{\left(\prod_{n=1}^{N} F_0(n)\right)^{\frac{1}{N}}}{\frac{1}{N}\sum_{n=1}^{N} F_0(n)} \tag{12}$$

$$VIB_S = \frac{\max_n (F_0(n))}{\sum_{n=1}^{N} F_0(n)} \tag{13}$$

Bad singers often try to use vibration of the sounds not for artistic purposes but to hide "false" intonation of a sound. In addition, false sounds are obviously directly showing lack of proficiency in vocal skills. Since intonation seems important in a voice quality determination, the base contour must be parametrized. In order to calculate intonation parameters, two methods were proposed. The first method calculates the medium value of a differential sequence of a base contour (IR). The second method does not analyze all base contour samples but the first and the last one, and returns the IT parameter. Parameters IR and IT are also defined for the first and last $N/2$ samples of pitch contour separately (N is the number of samples of the pitch contour) and are denoted as IR_{att}, IT_{att}, IR_{rel}, IT_{rel}, where*att* means the attack and *rel* the release of the sound.

3.4 Other Parameters

Another way of determining singing voice parameters is to use a more general signal description such as descriptors of audio content contained in the MPEG-7 standard. Although those parameters are not related to the singing voice biomechanics, they may be useful in the recognition process. The MPEG-7 parameters [11], [19] will not be presented in detail here, since they were reviewed in previous works by the authors [15], [17], [16]. The MPEG-7 audio parameters can be divided into the following groups:

- ASE (Audio Spectrum Envelope) describes the short-term power spectrum of the waveform. The mean values and variances of each coefficient over time are denoted as $ASE_1 \ldots ASE_{34}$ and $ASE_{1var} \ldots ASE_{34var}$ respectively.
- ASC (Audio Spectrum Centroid) describes the center of gravity of the log-frequency power spectrum. The mean value and the variance are denoted as ASC and ASC_{var} respectively.
- ASS (Audio Spectrum Spread). The mean value and the variance over time are denoted as ASS and ASS_{var} respectively.

– *SFM* (Spectral Flatness Measure) calculated for each frequency band. The mean values and the variances are denoted as $SFM_1 \ldots SFM_{24}$ and $SFM_{1var} \ldots SFM_{24var}$.
– Parameters related to discrete harmonic values: *HSD* (Harmonic Spectral Deviation), *HSS* (Harmonic Spectral Spread), *HSV* (Harmonic Spectral Variation).

The level of the first harmonic changes for different voice type qualities [27], thus in automatic voice recognition some parameters related to the behavior of harmonics can be useful. Parameters employed in the analysis were defined for harmonic decomposition of sounds. They are the mean value of differences between the amplitudes of a harmonic in adjacent time frames (s_n, where n is the number of a harmonic), the mean value of the amplitudes Ah of a harmonic over time (m_n, where n is the number of a harmonic), the standard deviation of amplitudes Ah of a harmonic over time (md_n, where n is the number of a harmonic).

Other parameters used in the experiments were: brightness (br) (center of spectrum gravity) [13], [14] and mel-cepstrum coefficients mcc_n [3], where n is the number of a coefficient.

4 Analysis of Parameters

All 2900 sound samples from the database described in Section 3 were described by the presented parameters. Since the total number of the parameters is big (331), they all will not all be listed here. We can, however, divide them into the following groups:

– parameters of formants – 46 parameters,
– parameters of the glottal - 59 parameters,
– parameters of the pitch contour (intonation and vibrato) – 18 parameters,
– other parameters (general) – 208 parameters.

4.1 Statistical Analysis

Some chosen pairs of the parameters can be represented graphically in a 2D space. In Fig. 2, an example of a distribution of two parameters for sound samples of professional and amateur singers is presented. It can be noticed, that for a majority of these samples sounds are separated by using only two features.

A large number of the features in the FV and a large number of voice samples are the reason to use statistical methods for the feature evaluation. Therefore every feature can be analyzed and a feature vector can be reduced to the parameters with the biggest values of statistics. Another way is to use the rough sets. Three methods of data reduction are to be described, namely Fisher statistic (F) and Sebestyen statistics (S), and rough sets in the following sections of this paper. Fisher statistic has the ability to test the separation between the pairs of classes being recognized, while Sebestyen criterion tests the database globally for

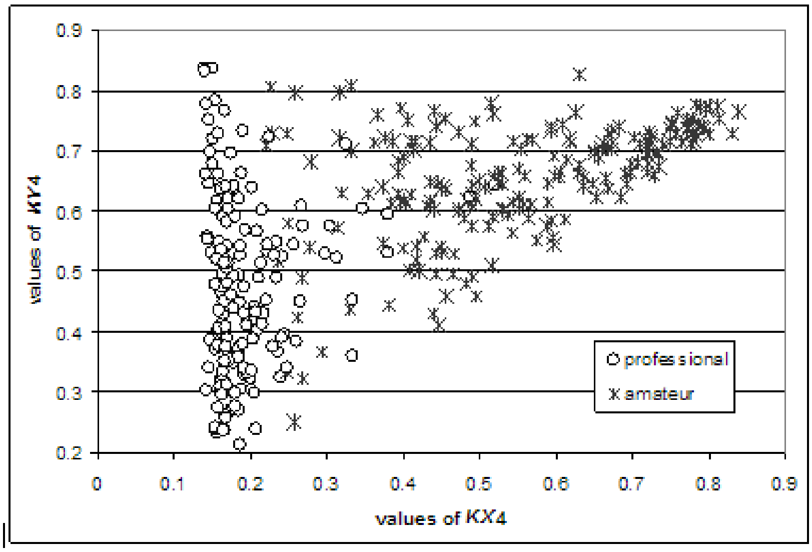


Fig. 2. An example of a 2D space of the values of selected parameters

Table 1. Sebestyen criterion for 20 parameters in the categories of voice quality (a), and voice type (b)

a.

parameter	Svalue	parameter	Svalue	parameter	Svalue
F_1/F_2	1.282	SFL_{min}	0.545	ASE_{15}	0.406
VIB_H	1.047	LAT	0.545	F_2/F_3	0.297
ASE_{16}	0.844	SFL_{med}	0.529	br	0.281
F_{31min}	0.672	ASE_{21}	0.519	F_{31med}	0.278
ASE_{23}	0.654	ASE_{22}	0.489	F_{22min}	0.252
ASE_{24}	0.637	SFL_{min}	0.468	F_{22max}	0.248
F_{22med}	0.556	ASE_{14}	0.407		

b.

parameter	Svalue	parameter	Svalue	parameter	Svalue
ASE_{10}	1.006	SFM_{17}	0.37	MCC_{10}	0.307
ASE_9	0.680	ASE_{25}	0.36	MCC_{10var}	0.307
LP_5	0.518	KP_4	0.358	F_{22min}	0.290
F_{22med}	0.509	$mfcc_{9var}$	0.358	ASE_{19}	0.258
ASE_{23}	0.501	ASE_{12}	0.355	MCC_8	0.258
ASE_{16}	0.419	LX_1	0.351	LP_6	0.241
MCC_6	0.384	ASE_{13}	0.320		

all pairs of classes in one calculation. Those statistical methods are presented in [13], [15], [26]. Sebestyen criterion has an advantage while compared to F statistic. Its global character enables to sort the parameters from the most to the less appropriate for all pairs of classes. In Table 1 the results of S criterion for 20 best parameters are presented in two categories of classification.

Fisher statistic allows for comparing parameters only in selected pairs of classes. Therefore the parameters cannot be globally compared. Below, are presented the most interesting conclusions coming from the statistical analysis of singing voice parameters that fall within the categories of the singing voice quality and type. The detailed studies of the parameter redundancy using the Fisher's criterion are presented in P. Zwan's PhD thesis [33].

In the category of voice quality, "dedicated" parameters obtained higher Fisher values than "general" parameters, while "general" descriptors were more appropriate for class separation in the voice type category. In the category of voice quality, the best BF results were obtained by glottal source parameters: LX_2, LX_3, SFL, VD, F_{22max}, F_1/F_2, F_2/F_3. Among "general" descriptors the best BF results were obtained by some chosen MPEG7 parameters: ASE and HSV (of various indexes) and the parameters describing value changes of harmonics in neighboring frames. For the pair of "amateur" – "professional" classes the best parameters (with regard to Fisher statistics) were the parameters related to the energy of the singer's formant: SFL, F_{22med}, F_1/F_3, F_2/F_3, F_{22min}. It is evident that the energy of the band of the singer's formant is crucial for distinguishing professional singers from amateurs. For the pair of: "semiprofessional" – "professional" classes the parameters related to the singer's formant energy do not have such a great significance. In this case, glottal source time parameters are essential. They relate to the invariability and periodicity of energy changes in singing to the level of single signal periods in voice samples. High values of the Fisher statistics were obtained by parameters related to vibrato: VD, VIB_P, VIB_H, VIB_S. Such a good result for vibrato parameters is very valuable, because these descriptors are not correlated with the parameters of the singer's formant (they describe different elements of singing technique).

In the category of voice type, the highest F values have threshold parameters KP_2, LP_4, LP_5, LP_8, parameters LX_1, KX_1, the SFL_{max} parameter related to the singer's formant level and the parameters related to a higher formant FW, namely: F_1/F_3. Among the parameters defined on the basis of the WLPC analysis, the highest F values were obtained by the parameters: F_{22med} and F_{22max}, what indicates the significance of defining the singer's formant values in relation to the second minima of the WLPC function.

The results of Sebestyen criterion and Fisher statistic cannot be compared directly, but generally the results are consistent. The majority of parameters with high S value also obtained high F value for a certain pair of classes, and similarly, the majority of parameters with a big Fisher criterion value had a high position in the list of parameters sorted by the S value. The consistence of the results proves the statistical methods to be good tools for a comparison of parameters in the context of their usability in the automatic recognition of singing voices.

4.2 Rough Set-Based Analysis

Rough sets introduced by Pawlak [21] are often employed in the analysis of data which aims to discover significant data and eliminate redundant ones. A reach literature on rough sets covers many applications [22], it is also used in music information retrieval [15], [28], [29], [29]. Within the context of this paper, the rough set method was used for the analysis of descriptors defined for the purpose of this study. In experiments, the rough set decision system RSES was employed [24]. Since this system is widely used by many researches, the details concerning its algorithmic implementation and performance will not be provided here. FVs were divided into training and testing sets. Parameters were quantized according to the RSES system principles. The local and global discretization were used to obtain reducts calculated from genetic and exhaustive algorithms [1]. Since two discretization methods and two algorithms for reduct calculation were used two sets of reduced parameters and four sets of reducts containing the selected parameters were extracted.

In the category of voice quality the vector of parameters was reduced to the parameters listed below:

a) the global discretization:

$$FV_1 = [F_{11},\ F_2/F_1,\ KX_2,\ KY_7,\ f_{VIB},\ VIB_p,\ ASE_{21},\ ASC,\ ASC_v,\ SFM_{10},\ s_2] \quad (15)$$

b) the local discretization:

$$FV_2 = [F_{11},\ F_{21},\ F_{31},\ F_{33},\ F_{12var},\ F_{13min},\ F_{13var},\ KX_1,\ KX_2,\ KP_1,\ LP_3,\ f_{VIB},\ VIB_p,\ LAT,\ TC,\ ASE_6,\ ASE_7,\ ASE_8,\ ASE_{21},\ s_2] \quad (16)$$

The sets of parameters selected by global and local discretization methods differ. The global discretization is a simpler method thus the number of parameters is lower. The global discretization tries to separate group of classes globally by selecting the lesser number of discretization cut points. However, the parameters chosen by the rough set method for both discretization methods in general match the parameters chosen by statistical methods of data reduction.

Among the reduced set of parameters, descriptors related to the WLPC analysis of formants can be found, and thus can be qualified as significant for the classification purposes. They are related to all three formants, which proves that in the category of voice quality all formants are required to be parameterized and the extracted descriptors should be contained in the FV. It is interesting that among those parameters F_{31} and F_{33}which are related to 'high formant' (middle frequency higher than 5kHz) appeared. The significance of this formant is not described in literature concerning automatic singing voice parametrization. Among glottal source parameters descriptors such as: KX_1, KX_2, KP_1, LP_3 were selected. On the other hand, frequency (f_{VIB}) and periodicity (VIB_p) related to vibrato modulation found their place among other important descriptors. From the remaining parameters, a few MPEG-7 parameters namely LAT, TC, ASE_6, ASE_7, ASE_8, ASE_{21} were qualified. In addition, one parameter related to the analysis of spectrum which is represented by s_2 related to the variation of the second harmonic was chosen.

In order to define the reducts, two algorithms were used: genetic and exhaustive algorithms. In the case of global discretization, those two algorithms calculated one and the same reduct containing all the parameters of (15). For both algorithms, all the parameters had equal influence on the decision. In the case of local discretization, reducts obtained by the two algorithms differed significantly. The resulting reducts are presented in (17) and (18). For selection of the number of the 'best' reducts the stability coefficient values was taken into account.

– reducts for the genetic algorithm, limited to a few best reducts:

$$\begin{array}{l}\{F_{11},\ F_{31},\ F_{12var},\ KX_2,\ f_{VIB},\ VIB_p,\ LAT,\ TC,\ ASE_6,\ ASE_7,\ ASE_8\}\\ \{F_{11},\ F_{31},\ F_{12var},\ F_{13min},KX_2,\ f_{VIB},VIB_p,\ LA,ASE_6,ASE_7,ASE_8,ASE_{21}\}\\ \{F_{11},\ F_{31},\ F_{13var},\ KX_2,\ f_{VIB},\ VIB_p,\ TC,\ ASE_6,\ ASE_7,\ ASE_8,\ ASE_{21}\}\\ \{F_{11},\ F_{31},\ KX_2,\ f_{VIB},\ VIB_p,\ LAT,\ TC,\ ASE_6,\ ASE_7,\ ASE_8,\ s_2\}\\ \{F_{11},\ F_{31},\ KX_2,\ f_{VIB},\ VIB_p,\ LAT,\ TC,\ ASE_6,\ ASE_7,\ ASE_8,\ ASE_{21}\}\\ \{F_{11},\ F_{13min},\ KX_2,\ KP_1,\ f_{VIB},\ VIB_p,TC,\ ASE_6,\ ASE_7,\ ASE_8,\ ASE_{18},\ \mathrm{s}_2\}\\ \{F_{31},\ F_{12var},\ KX_2,\ KP_1,\ f_{VIB},\ VIB_p,\ LAT,\ TC,\ ASE_6,\ ASE_7,\ ASE_8\}\\ \{F_{31},\ F_{13var},\ KX_2,\ f_{VIB},\ VIB_p,\ TC,\ ASE_6,\ ASE_7,\ ASE_8,\ ASE_{21},\ s_2\}\\ \{F_{13var},\ KX_2,\ KP_1,\ f_{VIB},\ VIB_p,\ TC,\ ASE_6,\ ASE_7,\ ASE_8,\ ASE_{21},\ s_2\}\\ \{F_{12var},\ KX_2,\ KP_1,\ f_{VIB},\ VIB_p,\ LAT,\ TC,\ ASE_6,\ ASE_7,\ ASE_8,\ ASE_{21}\}\\ \{F_{13min},\ KX_2,\ KP_1,\ f_{VIB},\ VIB_p,\ LAT,\ TC,\ ASE_6,\ ASE_7,\ ASE_8,\ s_2\}\end{array} \tag{17}$$

– reducts for the exhaustive algorithm, limited to best 6 reducts:

$$\begin{array}{l}\{KX_2,\ VIB_p,\ LAT,\ ASE_6,\ ASE_8\}\\ \{VIB_p,\ LAT,\ TC,\ ASE_6,\ ASE_8\}\\ \{F_{11},\ VIB_p,\ TC,\ ASE_6,\ ASE_8\}\\ \{F_{11},\ f_{VIB},\ VIB_p,\ ASE_6,\ ASE_8\}\\ \{KX_2,\ LAT,\ TC,\ ASE_6,\ ASE_8\}\\ \{KX_2,\ f_{VIB},\ VIB_p,ASE_6,\ ASE_8\}\end{array} \tag{18}$$

In the category of voice type over 200 from the total number of 331 parameters remained independently of the discretization methods or the type of algorithm used for the calculation. It was not possible to reduce parameter representation as much as in the case of the voice quality category. Within this context, automatic voice type recognition seems to be more complex. One of the reasons can be the diversity of registers among different voice types and individual voice qualities which change for singers for the same voice type. Additionally, some singers' voices were not easy to qualify to a voice type category, e.g. low registers of soprano voices were similar in timbre to mezzo-soprano and even alto voices.

5 Automatic Singing Voice Recognition

The next step of the experiment was to train an automatic recognition system based on both reduced and full feature vectors. Since three reduction methods were performed for each of the categories several decision systems (DS) were trained for the purposes of their comparison:

- DS 1 – for the full vector (331 parameters),
- DS 2 – for the vector with 100 parameters with biggest S values,
- DS 3 – for the vector with 100 parameters with biggest F values (all pairs of the classes were concerned),
- DS 4 – for the vector with 50 parameters with the biggest S values,
- DS 5 – for the vector witch 50 parameters with the biggest F values,
- DS 6 – for the vector with 20 parameters with the biggest S values,
- DS 7 – for the vector witch 20 parameters with the biggest F values,
- DS 8 – for the vector reduced by rough-sets with global discretization method
- DS 9 – for the vector reduced by rough-sets with local discretization method

Since Artificial Neural Networks are widely used in automatic sound recognition [13], [14], [15], [32], [35], the ANN classifier was used. The ANN was a simple feed-forward, three layer network with 100 neurons in the hidden layer and 3 or 6 neurons in the output layer respectively (dependent on the number of classes being recognized). Since there were 331 parameters in the FV, the input layer consisted of 331 neurons. Sounds from the database were divided into three groups. First part of samples (70%) was used for training, second part (10%) for validation and the third part (20%) for testing. Samples in training, validation and testing sets consisted of sounds of different vowels and pitches. The network was trained smoothly with the validation error increasing after approx. 3000 cycles. To train the network optimally, the minimum of the global validation error function must have been found. If the validation error was increasing for 50 successive cycles, the last validation error function minimum was assumed to be global, and the learning was halted. In Figure 3, the automatic recognition results are presented for nine decision systems DS1 – DS9 and two recognition categories. V_{331} is the vector of all 331 parameters, S_n are the vectors reduced by the Sebestyen criterion to n parameters, F_m are the vectors reduced by the Fisher statistic to m parameters, RS_L is the vector of parameters selected by the rough set local discretization method and RS_G is the vector of parameters selected by the rough set global discretization method.

The results from Table 2 show that whatever the data reduction method is, artificial neural networks generate similar results of automatic recognition. The efficiency for a full vector of 331 parameters is 93.4% for the voice quality category and 90% for the voice type category and decreases to approx. 75% when the size of the vector is reduced to 20 parameters. A better recognition accuracy for the voice type category when the rough set data reduction method is used comes from the fact that for this category the vector was not significantly reduced. In the case of the voice quality the results of automatic recognition for F_{20}, S_{20} and RS_L can directly be compared because in those two cases the

Table 2. Results of automatic recognition accuracy [%] for various FV size reduction methods

Category	V_{331}	S_{100}	F_{100}	S_{50}	F_{50}	S_{20}	F_{20}	RS_L	RS_G
Quality	93.4	90.5	92.1	85.3	87.2	**73.2**	**75.5**	**75.2**	62
Type	90.0	82.3	81.5	79.0	76.0	58.3	60.1	83.1	82.5

Table 3. The comparison of the accuracy of RS and ANN based classifiers for various RS-based data reduction methods

Data reduction method	RS [%]	ANN [%]
global discretization, both algorithms	96.8	62
local discretization, genetic algorithm	97.6	72.5
local discretization, exhaustive algorithm	89	72.5

vectors have the same number of parameters. The recognition accuracy is very similar for all three methods.

Rough set-based algorithms can serve not only for data reduction purposes but also as classifiers. A comparison between RS and ANN classifiers acting on vectors reduced by rough set-based methods seems very interesting. Since the reducts were extracted only for the vocal quality category, the experiment was carried on for that category and the results are presented in Table 3.

The automatic recognition results are much better for an RS classifier used. The RS method is specialized for the classification task of a reduced set of parameters. A discretization algorithm used in RS selects the most important cut points in terms of the discernibility between classes. Thus, rules generated from the parameters by RS are strictly dedicated for the analyzed case. Following the RS methodology, the proper rules are easy to obtain. For ANNs, since they are trained and tested using single objects, the generalization is harder to obtain and every single training object can have an influence on the results. In the case of a smaller number of parameters it has a particular meaning which can be clearly observed in Table 3. Contrarily, when the number of parameters (the number of reducts) is bigger, the ANN decision system starts to perform better than RS. This may be observed in the results of the automatic recognition in the voice type category (parts of Tables 4 and 5).

In order to make a detailed comparison, between the best trained ANN recognition system and the best trained RSES system, the detailed recognition results for both recognition categories are presented in Tables 4 and 5. Rows in these tables describe recognized quality classes, and columns correspond to the classification.

In the case of the quality category, the automatic recognition results are better comparing to the ANN. The rough set system achieved very good results with a reduced FV of 20 parameters in the classification of the voice quality category. In the category of voice type, the results are worse. Moreover, in the case of the voice type category erroneous classification is not always related to neighboring

Table 4. ANN singing voice recognition results for (a) Voice Quality (VQ) and (b) Voice Type (VT) categories

a.

VQ recognition [%]	amateur	semi-professional	professional
amateur	96.3	2.8	0.9
semi-professional	4.5	94.3	1.1
professional	3.5	7	89.5

b.

VT category recognition [%]	bass	baritone	tenor	alto	mezzo	soprano
bass	90.6	6.3	3.1	0	0	0
baritone	3.3	90	6.7	0	0	0
tenor	0	3.6	89.3	7.1	0	0
alto	0	0	4	80	12	4
mezzo	0	0	0	0	93.8	6.3
soprano	0	0	2.9	0	2.9	94.1

Table 5. RSES-based singing voice classification results for (a) Voice Quality (VQ) and (b) Voice Type (VT) categories

a.

VQ recognition [%]	amateur	semi-professional	professional
amateur	94.7	4.2	1.1
semi-professional	1.3	95.4	3.3
professional	0	1.6	96.7

b.

VT category recognition [%]	bass	baritone	tenor	alto	mezzo	soprano
bass	84.0	10.0	4.0	2.0	0	0
baritone	13.0	64.8	13.0	0	1.9	7.3
tenor	6.0	18.0	54.0	10.0	6.0	6.0
alto	0	4.7	16.3	51.2	16.3	11.6
mezzo	3.8	0	2.6	1.3	73.1	19.2
soprano	2.9	2.9	2.9	1.4	11.4	78.6

classes. Thus, the RSES system was not able to perform the classification as well as ANN while trained and tested on vectors of more than 200 parameters in the category of voice type where further vector size reduction was not possible (total accuracy obtained equals 0.664). It is interesting to notice that types of voices being 'the extreme' of the voice type category were recognized with better efficiency than those contained between other classes. Also, there is not much difference whether this concerns male or female voice.

6 Conclusions

By comparing automatic recognition results of neural networks and rough set systems, several conclusions may be reached. The recognition performed by the rough set system was better for the quality category and worse for the voice type category in comparison to the ANN. In the case of the voice quality category, it was possible for the RS system to reduce a large number of parameters to 20 descriptors and the extraction of rules went very smoothly. Descriptors of the level of formants, stability of glottal parameters along with those related to vibrato, and MPEG-7 descriptors in addition, enabled to derive linear IF-THEN rules. It proves that automatic recognition of the quality category is possible for a significantly reduced number of descriptors.

In the case of voice quality it was not possible to achieve very good recognition results for the RS classifier as the extraction of a small number of rules was not possible. Neural networks enabled to classify particular types of singing voices effectively while the rough-set system achieved worse efficiency. The diversity of voice registers and individual timbre characteristics of singers are the reason that non-linear classification systems such as ANNs should perhaps be used for automatic recognition in the category of voice type. Another reason for lower recognition results may be that the database of singing voices was represented by too few different singers.

Moreover, it has been proven that all the presented data reduction algorithms enabled a significant decrease in the feature vector size. The results obtained by the trained ANN for vectors of the same length but produced by different data reduction methods gave very similar recognition results. The parameters selected by those algorithms as the most appropriate for automatic singing voice recognition were very similar.

Acknowledgments. The research was partially supported by the Polish Ministry of Science and Education within the project No. PBZ-MNiSzW-02/II/2007.

References

1. Bazan, J.G., Szczuka, M.S.: The Rough Set Exploration System. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets III. LNCS, vol. 3400, pp. 37–56. Springer, Heidelberg (2005)
2. Bloothoof, G.: The sound level of the singers formant in professional singing. J. Acoust. Soc. Am. 79(6), 2028–2032 (1986)
3. Childers, D.G., Skinner, D.P., Kemerait, R.C.: The Cepstrum: A Guide to Processing. Proc. IEEE 65, 1428–1443 (1977)
4. Dejonckere, P.H., Olek, M.P.: Exactness of intervals in singing voice: A comparison between singing students and professional singers. In: Proc. 17th International Congress on Acoustics, Rome, VIII, pp. 120–121 (2001)
5. Diaz, J.A., Rothman, H.B.: Acoustic parameters for determining the differences between good and poor vibrato in singing. In: Proc. 17th International Congress on Acoustics, Rome, VIII, pp. 110–116 (2001)

6. Dziubiński, M., Kostek, B.: Octave Error Immune and Instantaneous Pitch Detection Algorithm. J. of New Music Research 34, 273–292 (2005)
7. Fry, D.B.: Basis for the acoustical study of singing. J. Acoust. Soc. Am. 28, 789–798 (1957)
8. Harma, A.: Evaluation of a warped linear predictive coding scheme. In: Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 2, pp. 897–900 (2000)
9. Harma, A.: A comparison of warped and conventional linear predictive coding. IEEE Transactions on Speech and Audio Processing 5, 579–588 (2001)
10. Herzel, H., Titze, I., Steinecke, I.: Nonlinear dynamics of the voice: signal analysis and biomechanical modeling. CHAOS 5, 30–34 (1995)
11. Herrera, P., Serra, X., Peeters, G.: A proposal for the description of audio in the context of MPEG-7. In: Proc. CBMI European Workshop on Content-Based Multimedia Indexing, Toulouse, France (1999)
12. Joliveau, E., Smith, J., Wolfe, J.: Vocal tract resonances in singing: the soprano voice. J. Acoust. Soc. America 116, 2434–2439 (2004)
13. Kostek, B.: Soft Computing in Acoustics, Applications of Neural Networks, Fuzzy Logic and Rough Sets to Music Acoustics, Studies in Fuzziness and Soft Computing. Physica Verlag, Heidelberg (1999)
14. Kostek, B., Czyżewski, A.: Representing Musical Instrument Sounds for Their Automatic Classification. J. Audio Eng. Soc. 49, 768–785 (2001)
15. Kostek, B.: Perception-Based Data Processing in Acoustics. In: Applications to Music Information Retrieval and Psychophysiology of Hearing. Series on Cognitive Technologies. Springer, Heidelberg (2005)
16. Kostek, B., Szczuko, P., Żwan, P., Dalka, P.: Processing of Musical Data Employing Rough Sets and Artificial Neural Networks. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets III. LNCS, vol. 3400, pp. 112–133. Springer, Heidelberg (2005)
17. Kostek, B.: Applying computational intelligence to musical acoustics. Archives of Acoustics 32(3), 617–629 (2007)
18. Kruger, E., Strube, H.W.: Linear prediction on a warped frequency scale. IEEE Trans. on Acoustics, Speech, and Signal Processing 36(9), 1529–1531 (1988)
19. Lindsay, A., Herre, J.: MPEG-7 and MPEG-7 Audio - An Overview. J. Audio Eng. Society 49(7/8), 589–594 (2001)
20. Mendes, A.: Acoustic effect of vocal training. In: Proc. 17th International Congress on Acoustics, Rome, VIII, pp. 106–107 (2001)
21. Pawlak, Z.: Rough Sets. International J. Computer and Information Sciences 11, 341–356 (1982)
22. Peters, J.F., Skowron, A. (eds.): Transactions on Rough Sets V. LNCS, vol. 4100. Springer, Heidelberg (2006)
23. Rabiner, L.: On the use of autocorrelation analysis for pitch detection. IEEE Trans., ASSP 25, 24–33 (1977)
24. Rough-set Exploration System, `logic.mimuw.edu.pl/~rses/RSES_doc_eng.pdf`
25. Schutte, H.K., Miller, D.G.: Acoustic Details of Vibrato Cycle in Tenor High Notes. J. of Voice 5, 217–231 (1990)
26. Sebestyen, G.S.: Decision-making processes in pattern recognition. Macmillan Publishing Co., Indianapolis (1965)
27. Sundberg, J.: The science of the singing voice. Northern Illinois University Press (1987)
28. Wieczorkowska, A., Czyżewski, A.: Rough Set Based Automatic Classification of Musical Instrument Sounds. Electr. Notes Theor. Comput. Sci. 82(4) (2003)

29. Wieczorkowska, A., Raś, Z.W.: Editorial: Music Information Retrieval. J. Intell. Inf. Syst. 21(1), 5–8 (2003)
30. Wieczorkowska, A., Ras, Z.W., Zhang, X., Lewis, R.A.: Multi-way Hierarchic Classification of Musical Instrument Sounds, pp. 897–902. MUE, IEEE (2007)
31. Wolf, S.K.: Quantitative studies on the singing voice. J. Acoust. Soc. Am. 6, 255–266 (1935)
32. Żwan, P.: Expert System for Automatic Classification and Quality Assessment of Singing Voices. 121 Audio Eng. Soc. Convention, San Francisco, USA (2006)
33. Żwan, P.: Expert system for objectivization of judgments of singing voices (in Polish), Ph.D. Thesis (supervisor: Kostek B.), Gdansk Univ. of Technology, Electronics, Telecommunications and Informatics Faculty, Multimedia Systems Department, Gdansk, Poland (2007)
34. Żwan, P., Kostek, B., Szczuko, P., Czyżewski, A.: Automatic Singing Voice Recognition Employing Neural Networks and Rough Sets. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 793–802. Springer, Heidelberg (2007)
35. Żwan, P.: Automatic singing quality recognition employing artificial neural networks. Archives of Acoustics 33(1), 65–71 (2008)

Hierarchical Classifiers for Complex Spatio-temporal Concepts

Jan G. Bazan

Chair of Computer Science, University of Rzeszów
Rejtana 16A, 35-310 Rzeszów, Poland
bazan@univ.rzeszow.pl

Abstract. The aim of the paper is to present rough set methods of constructing hierarchical classifiers for approximation of complex concepts. Classifiers are constructed on the basis of experimental data sets and domain knowledge that are mainly represented by concept ontology. Information systems, decision tables and decision rules are basic tools for modeling and constructing such classifiers. The general methodology presented here is applied to approximate spatial complex concepts and spatio-temporal complex concepts defined for (un)structured complex objects, to identify the behavioral patterns of complex objects, and to the automated behavior planning for such objects when the states of objects are represented by spatio-temporal concepts requiring approximation. We describe the results of computer experiments performed on real-life data sets from a vehicular traffic simulator and on medical data concerning the infant respiratory failure.

Keywords: rough set, concept approximation, complex dynamical system, ontology of concepts, behavioral pattern identification, automated planning.

1 Introduction

Reasoning based on concepts constitutes one of the main elements of a thinking process because it is closely related to the skill of categorization and classification of objects. The term *concept* means *mental picture of a group of objects* (see [1]). While the term *conceptualize* is commonly understood to mean *form a concept or idea about something* (see [1]). In the context of this work, there is interest in classifying conceptualized sets of objects. Concepts themselves provide a means of *describing* (forming a mental picture of) sets of objects (for a similar understanding the term *concept*, see, *e.g.*, [2, 3, 4]).

Definability of concepts is a term well-known in classical logic (see, *e.g.*, [5]). Yet in numerous applications, the concepts of interest may only be defined approximately on the basis of available, incomplete information about them (represented, *e.g.*, by positive and negative examples) and selected primary concepts and methods for creating new concepts out of them. It brings about the necessity to work out approximate reasoning methods based on inductive reasoning (see, *e.g.*, [6, 7, 8, 9, 10, 11, 12, 13]).

In machine learning, this issue is known under the term *learning concepts by examples* (see, *e.g.*, [10]). The main problem of learning concepts by examples is that the description of a concept under examination needs to be created on the basis of known examples of that concept. By creating a concept description we understand detection of such properties of exemplary objects belonging to this concept that enable further examination of examples in terms of their membership in the concept under examination. A natural way to solve the problem of learning concepts by examples is inductive reasoning which means that while obtaining further examples of objects belonging to the concept (the so-called positive examples) and examples of objects not belonging to the concept (the so-called negative examples), an attempt is made to find such a description that correctly matches all or almost all examples of the concept under examination. Moreover, instead of speaking of learning concepts by examples, one may consider a more general learning of the so-called classifications which are partitions of all examples into a family of concepts (called decision classes) creating a partition of the object universe. A description of such a classification makes it possible to recognize the decision that should be made about examples unknown so far; that is, it gives us the answer as to what decision should be made that also includes examples not occurring in the process of classification learning.

Classifiers also known in literature as *decision algorithms*, *classifying algorithms* or *learning algorithms* may be treated as constructive, approximate descriptions of concepts (decision classes). These algorithms constitute the kernel of *decision systems* that are widely applied in solving many problems occurring in such domains as *pattern recognition*, *machine learning*, *expert systems*, *data mining* and *knowledge discovery* (see, *e.g.*, [6, 8, 9, 10, 11, 12, 13]).

In literature there can be found descriptions of numerous approaches to constructing classifiers, which are based on such paradigms of machine learning theory as *classical and modern statistical methods* (see, *e.g.*, [11, 13]), *neural networks* (see, *e.g.*, [11, 13]), *decision trees* (see, *e.g.*, [11]), *decision rules* (see, *e.g.*, [10, 11]), and *inductive logic programming* (see, *e.g.*, [11]). Many of the approaches mentioned above resulted in decision systems intended for computer support of decision making (see, *e.g.*, [11]). An example of such a system is RSES (*Rough Set Exploration System* [14, 15]) which has been developed for over ten years and utilizes rough set theory, originated by Professor Zdzisław Pawlak (see [16, 17, 18]), in combination with Boolean reasoning (see [19, 20, 21]).

With the development of modern civilization, not only the scale of the data gathered but also the complexity of concepts and phenomena which they concern are increasing rapidly. This crucial data change has brought new challenges to work out new data mining methods. Particularly, data more and more often concerns complex processes which do not give in to classical modeling methods. Of such a form may be medical and financial data, data coming from vehicles monitoring, or data about the users gathered on the Internet. Exploration methods of such data are in the center of attention in many powerful research centers in the world, and at the same time detection of models of complex processes and their properties (patterns) from data is becoming more and more attractive for applications (see, *e.g.*, [22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]). Making a progress in this field is extremely crucial, among other things, for the development of intelligent systems which support decision making on the basis of results of analysis of the available data sets. Therefore, working out methods of detection of process models and their properties from data and proving their effectiveness in different applications are of particular importance for the further development of decision supporting systems in many domains such as medicine, finance, industry, transport, telecommunication, and others.

However, in the last few years essential limitations have been discovered concerning the existing data mining methods for very large data sets regarding complex concepts, phenomena, or processes (see, *e.g.*, [41, 42, 43, 44, 45, 46]). A crucial limitation of the existing methods is, among other things, the fact that they do not support an effective approximation of complex concepts, that is, concepts whose approximation requires discovery of extremely complex patterns. Intuitively, such concepts are too far in the semantical sense from the available concepts, *e.g.*, sensory ones. As a consequence, the size of spaces which should be searched in order to find patterns crucial for approximation are so large that an effective search of these spaces very often becomes unfeasible using the existing methods and technology. Thus, as it turned out, the ambition to approximate complex concepts with high quality from available concepts (most often defined by sensor data) in a fully automatic way, realized by the existing systems and by most systems under construction, is a serious obstacle since the classifiers obtained are often of unsatisfactory quality.

Recently, it has been noticed in the literature (see, *e.g.*, [42, 47, 48, 49, 50, 51, 52]) that one of the challenges for data mining is discovery of methods linking detection of patterns and concepts with domain knowledge. The latter term denotes knowledge about concepts occurring in a given domain and various relations among them. This knowledge greatly exceeds the knowledge gathered in data sets; it is often represented in a natural language and usually acquired during a dialogue with an expert in a given domain. One of the ways to represent domain knowledge is to record it in the form of the so-called *concept ontology* where ontology is usually understood as a finite hierarchy of concepts and relations among them, linking concepts from different levels (see, *e.g.*, [53, 54]).

In the paper, we discuss methods for approximation of complex concepts in real-life projects. The reported research is closely related to such areas as *machine learning* and *data mining* (*feature selection and extraction* [55, 56, 57], *classifier construction* [9, 10, 11, 12], *analytical learning* and *explanation based learning* [12, 58, 59, 60, 61]), *temporal and spatio-temporal reasoning* [62, 63, 64], *hierarchical learning and modeling* [42, 52, 65, 66, 67, 68], *adaptive control* [67, 69], *automated planning* (hierarchical planning, reconstruction of plans, adaptive learning plans) [70, 71, 72, 73, 74, 75, 76], *rough sets* and *fuzzy sets* (*approximation of complex vague concepts*) [77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87], *granular computing* (searching for compound patterns) [88, 89, 90, 91], *complex adaptive systems* [92, 93, 94, 95, 96, 97], *autonomous multiagent systems* [98, 99, 100, 101]), *swarm systems* [102, 103, 104], *ontologies development* [53, 54, 105, 106, 107].

It is also worthwhile mentioning that the reported research is also closely related to the domain of clinical decision-support for medical diagnosis and therapy (see, *e.g.*, [108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121]). Many reported results in this domain can be characterized as methods for solving specific problems such as *temporal abstraction problem* [117, 120, 121] or *medical planning problem* [108, 111, 112, 119]). Many methods and algorithms proposed in this paper can be also used for solving such problems.

The main aim of the paper is to present the developed methods for approximation of complex vague concepts involved in specification of real-life problems and approximate reasoning used in solving these problems. However, methods presented in the paper are assuming that additional domain knowledge in the form of the concept ontology is given. Concepts from ontology are often vague and expressed in natural language. Approximation of ontology is used to create hints in searching for approximation of complex concepts from sensory (low level) data.

The need of use of a domain knowledge expressed in the form of a concept ontology can be noticed in intensively developing domains connected with analysis and data processing as in the case of *reinforcement learning* (see, *e.g.*, [12, 122, 123, 124]). In the latter field, methods of learning new strategies with reinforcement take into account concept ontologies obtained from an expert, with the help of which it is possible to construct an approximation of a function estimating the quality of actions performed. Similarly, in a *Service Oriented Architecture* (SOA) [47, 49], the distribution of varied Web Services can be performed with the use of a domain knowledge, expressed using a concept ontology.

There also appeared propositions (see, *e.g.*, [42, 51, 52]) that use domain knowledge to search for the approximation of complex concepts in a hierarchical way which would lead to hierarchical classifiers able to approximate complex concepts with the high quality, *e.g.*, by analogy to biological systems [42]. This idea can be also related to learning of complex (*e.g.*, nonlinear) functions for fusion of information from different sources [125]. Therefore, currently, the problem of construction of such hierarchical classifiers is fundamental for complex concepts approximation and its solution will be crucial for construction of many methods of intelligent data analysis. These are, for example,

– methods of classification of objects into *complex spatial concepts* which are semantically distant from sensor data, *e.g.*, these are concepts as *safe vehicle driving on a highway*, *hazardous arrangement of two cooperating robots which puts them both at risk of being damaged*,
– methods of classification of object to *complex spatio-temporal concepts* semantically distant from sensor data which require observation of single objects or many objects over a certain period of time (*e.g.*, *acceleration of a vehicle on the road*, *gradual decrease of a patient's body temperature*, *robot's backward movement while turning right*),
– methods of *behavioral pattern* or *high risk pattern* identification where these types of patterns may be treated as complex concepts representing dynamic properties of objects; such concepts are expressed in a natural language on a

high level of abstraction and describing specific behaviors of a single object (or many complex objects) over a certain period of time (*e.g.*, *overtaking one vehicle by another*, *a traffic jam*, *chasing one vehicle after another*, *behavior of a patient under a high life threat*, *ineffective cooperation of a robot team*)
– methods of automatic learning of plans of complex object behavior, where a plan may be treated as a complex value of the decision which needs to be made for complex objects such as *vehicles*, *robots*, *groups of vehicles*, *teams of robots*, or *patients undergoing treatment.*

In the paper, we propose to link automatic methods of complex concept learning, and models of detection of processes and their properties with domain knowledge obtained in a dialogue with an expert. Interaction with a domain expert facilitates guiding the process of discovery of patterns and models of processes and makes the process computationally feasible. Thus presentation of new approximation methods of complex concepts based on experimental data and domain knowledge, represented using ontology concepts, is the main aim of this paper. In our opinion, the presented methods are useful for solving typical problems appearing when modeling complex dynamical systems.

1.1 Complex Dynamical Systems

When modeling complex real-world phenomena and processes mentioned above and solving problems under conditions that require an access to various distributed data and knowledge sources, the so-called *complex dynamical systems* (CDS) are often applied (see, *e.g.*, [92, 93, 94, 95, 96, 97]), or putting it in other way *autonomous multiagent systems* (see, *e.g.*, [98, 99, 100, 101]) or *swarm systems* (see, *e.g.*, [104]). These are collections of complex interacting objects characterized by constant change of parameters of their components over time, numerous relationships between the objects, the possibility of cooperation/competition among the objects and the ability of objects to perform more or less compound actions. Examples of such systems are *traffic*, *a patient observed during treatment*, *a team of robots during performing some task*, etc.

It is also worthwhile mentioning that the description of a CDS dynamics is often not possible with purely analytical methods as it includes many complex vague concepts (see, *e.g.*, [126, 127, 128]). Such concepts concern properties of chosen fragments of the CDS and may be treated as more or less complex objects occurring in the CDS. Hence, are needed appropriate methods of extracting such fragments that are sufficient to conclude about the global state of the CDS in the context of the analyzed types of changes and behaviors. In this approach, the CDS state is described by providing information about the membership of the complex objects isolated from the CDS in the complex concepts already established, describing properties of complex objects and relations among these objects. Apart from that, the description of the CDS dynamics requires following changes of the CDS state in time which leads to the so-called *trajectory* (history), that is, sequences of the CDS states observed over a certain period of time. Therefore, there are also needed methods for following changes of the selected fragments of the CDS and changes of relations between the extracted fragments. In this paper, we use complex spatio-temporal concepts concerning properties, describing the dynamics of complex objects occurring in CDSs, to represent and monitor such changes. They are expressed in natural language on a much higher level of abstraction than so-called sensor data, so far mostly applied to the approximation of concepts. Examples of such concepts are *safe car driving*, *safe overtaking*, *patient's behavior when faced with a life threat*, *ineffective behavior of robot team.*

However, the identification of complex spatio-temporal concepts and using them to monitor a CDS requires approximation of these concepts. In this paper, we propose to approximate complex spatio-temporal concepts by hierarchical classifiers mentioned above and based on data sets and domain knowledge.

1.2 Problems in Modeling Complex Dynamical Systems

In modeling complex dynamical systems there appear many problems related to approximation of complex concepts used to describe the dynamics of the systems. One of these problems is obviously the problem of the gap between complex concepts and sensor data mentioned above. Apart from that, a series of other problems may be formulated whose solution is very important for complex concepts approximation and for complex dynamical systems monitoring. Below, we present a list of such problems including particularly those whose solution is the aim of this paper.

1. Problem of the gap between complex concepts and sensor data preventing an effective direct usage of sensor data to induce approximation of complex concepts by fully automatic methods.
2. Problem of complex concept stratification in classifier construction.
3. Problem of identification of behavioral patterns of complex objects in complex dynamical systems monitoring.
4. Problem of context of complex object parts while complex dynamical systems monitoring.
5. Problem of time speed-up in identification of behavioral patterns.
6. Problem of automated planning of complex object behavior when the object states are represented by complex concepts requiring approximation.
7. Problem of solving conflicts between actions in automated planning of complex object behavior.
8. Problem of synchronization of plans constructed for parts of a structured complex object.
9. Problem of plan adaptation.
10. Problem of similarity relation approximation between complex objects, complex object states, and complex object behavioral plans using data sets and domain knowledge.

In further subsections, a brief overview of the problems mentioned above is presented.

Problem of the Gap between Complex Concepts and Sensor Data. As we mentioned before, in spatio-temporal complex concepts approximation using sensor data, there occur major difficulties resulting from the fact that between spatio-temporal complex concepts and sensor data, there exists a gap which prevents an effective direct usage of sensor data for approximation of complex concepts. Therefore, in the paper we propose to fill the gap using domain knowledge represented mainly by a concept ontology and data sets chosen appropriately for this ontology (see Section 1.3).

Problem of Complex Concept Stratification. When we create classifiers for concepts on the basis of uncertain and imprecise data and knowledge semantically distant from the concepts under approximation, it is frequently not possible to construct a classifier which decisively classifies objects, unknown during classifier learning, to the concept or its complement. There appears a need to construct such classifiers that, instead of stating clearly about the object under testing whether it belongs to the concept or not, allow us to obtain only a certain type of membership degree of the object under testing to the concept. In other words, we would like to determine, with regards to the object under testing, how certain the fact that this object belongs to the concept is. Let us notice that this type of mechanism stratifies concepts under approximation, that is, divides objects under testing into layers labeled with individual values of membership degree to the concept. Such a mechanism can be obtained using different kinds of probability distributions (see [6, 43]). However, in this paper, instead of learning of a probability distribution we learn layers of concepts relevant for construction of classifiers. We call such classifiers as stratifying classifiers and we present two methods of a stratifying classifier construction (see Section 1.3). Our approach is inspired by papers about linguistic variables written by Professor Lotfi Zadeh (see [129, 130, 131]).

Problem of Identifying Behavioral Patterns. The study of collective behavior in complex dynamical systems is now one of the more challenging research problems (see, *e.g.*, [93, 99, 100, 102, 104, 132, 133, 134]), especially if one considers the introduction of some form of learning by cooperating agents (see, *e.g.*, [103, 122, 123, 124, 135, 136, 137]). For example, an efficient complex dynamical systems monitoring very often requires the identification of the so-called *behavioral patterns* or a specific type of such patterns called *high-risk patterns* or *emergent patterns* (see, *e.g.*, [93, 99, 100, 132, 138, 139, 140, 141, 142, 143, 144]). They are complex concepts concerning dynamic properties of complex objects expressed in a natural language on a high level of abstraction and describing specific behaviors of these objects. Examples of behavioral patterns may be: *overtaking one vehicle by another vehicle*, *driving a group of vehicles in a traffic jam*, *behavior of a patient under a high life threat*, etc. These types of concepts are difficult to identify automatically because they require watching complex object behavior over longer period of time and this watching usually is based on the identification of a sequence of less complex spatio-temporal concepts. Moreover, a crucial role for identification of a given behavioral pattern is played by the sequence of less complex concepts which identify it. For example, in order to identify the behavioral pattern of *overtaking one vehicle by another*, it should first be determined whether the overtaking vehicle approaches the overtaken vehicle; next, whether the overtaking vehicle changes lanes appropriately and overtakes the vehicle; and finally, to determine that the overtaking vehicle returns to the previous lane driving in front of the overtaken vehicle. The methodology of a dynamical system modeling proposed in the paper enables approximation of behavioral patterns on the basis of data sets and domain knowledge expressed using a concept ontology (see Section 1.3).

Problem of Context for Complex Object Parts. In this paper, any complex dynamical system (CDS) is represented using descriptions of its global states or trajectories (histories), that is, sequences of CDS states observed over a certain period of time (see, *e.g.*, [145, 146, 147, 148, 149, 150, 151, 152] and Section 1.1). Properties of such states or trajectories are often dependent on specific parts of these states or trajectories. This requires to consider the relevant structure of states or trajectories making it possible to extract parts and the relevant context of parts. Moreover, each structured object occurring in a complex dynamical system is understood as a set of parts extracted from states or trajectories of a given complex dynamical system. Such parts are often related by relations representing links or interactions between parts. That is why both learning of the behavioral patterns concerning structured objects and the identification of such patterns, in relation to specific structured objects, requires the isolation of structured objects as sets of potential parts of such objects, that is, object sets of lesser complexity. The elementary approach to isolate structured objects consisting in examination of all possible subsets (of an established size) of the set of potential parts of structured objects cannot be applied because of potentially high number of such subsets. For example, during an observation of a highway from a helicopter (see, *e.g.*, [89, 153]), in order to identify a group of vehicles which are involved in the maneuver of dangerous overtaking, it would be necessary to follow (in the real time) the behavior of all possible groups of vehicles of an established size (*e.g.*, six vehicles, see Appendix A) that may be involved in this maneuver, which already with a relatively small number of visible vehicles becomes computationally too difficult.

Another possibility is the application of methods which use the context in which the objects being parts of structured objects occur. This type of methods isolate structured objects not by a direct indication of the set of parts of the searched structured object but by establishing one part of the searched structured object and attaching to it other parts, being in the same context as the established part. Unfortunately, also here, the elementary approach to determination of the context of the part of the structured object, consisting in examination of all possible subsets (of an established size) of the set of potential structured objects to which the established part of the structured object belongs, cannot be applied because of a large number of such subsets. For example, in order to identify a group of vehicles which are involved in a dangerous maneuver and to which the vehicle under observation belongs, it would be necessary to follow (in the real time) the behavior of the possible groups of vehicles of an established size (*e.g.*, six vehicles, see Appendix A) to which the vehicle considered belongs, which is, with a relatively small number of visible vehicles, still computationally too difficult. Therefore, there are needed special methods of determining the context of the established part of the structured object based on a domain knowledge which enable to limit the number of analyzed sets of parts of structured objects. In the paper, we propose the so-called *sweeping method* which enables fast determination of the context of the established object treated as one of the parts of the structured object (see Section 1.3).

Problem of Time Speed-Up in Identification of Behavioral Patterns. Identification of a behavioral pattern in relation to a specific complex object may be performed by observing the behavior of these objects over a certain period of time. Attempts to shorten this time are usually inadvisable, because they may cause false identification of behavioral pattern in relation to some complex objects. However, in many applications there exists a need for a fast decision making (often in the real time) about whether or not a given object matches the established behavioral pattern. It is extremely crucial in terms of computational complexity because it enables a rapid elimination of these complex objects which certainly do not match the pattern. Therefore, in the paper, there is presented a method of elimination of complex objects in identification of a behavioral pattern, which is based on *the rules of fast elimination of behavioral patterns* which are determined on the basis of data sets and domain knowledge (see Section 1.3).

Problem of Automated Planning. In monitoring the behavior of complex dynamical systems (*e.g.*, by means of behavioral patterns identification) there may appear a need to apply methods of automated planning of complex object behavior. For example, if during observation of a complex dynamical system, a behavioral pattern describing inconvenient or unsafe behavior of a complex object (i.e., a part of system state or trajectory) is identified, then the system control module may try, using appropriate actions, to change the behavior of this object in such a way as to lead the object out of the inconvenient or unsafe situation. However, this type of short-term interventions may not be sufficient to lead the object out of the undesired situation permanently. Therefore, a possibility of automated planning is often considered which means construction of sequences of actions alternately with states (of plans) to be performed by the complex object or on the complex object in order to bring it to a specific state. In literature, there may be found descriptions of many automated planning methods (see, *e.g.*, [70, 71, 72, 73, 74, 75, 76]). However, applying the latter approaches, it has to be assumed that the current complex object state is known which results from a simple analysis of current values of available parameters of this object. Meanwhile, in complex dynamical systems, a complex object state is often described in a natural language using vague spatio-temporal conditions whose satisfiability cannot be tested on the basis of a simple analysis of available information about the object. For example, when planning the treatment of an infant suffering from the respiratory failure, the infant's condition may be described by the following condition:

– *Patient with RDS type IV, persistent PDA and sepsis with mild internal organs involvement* (see Appendix B for mor medical details).

Stating the fact that a given patient is in the above condition requires an analysis of examination results of this patient registered over a certain period of time with a large support of a domain knowledge provided by experts (medical doctors). This type of conditions may be represented using complex spatio-temporal concepts. Identification of these conditions requires, however, an approximation of the concepts representing them with the help of classifiers. Therefore, in the paper, we describe automated planning methods of behavior of complex objects whose states are described using complex concepts requiring approximation (see Section 1.3).

Problem of Solving Conflicts between Actions. In automated planning methods, during a plan construction there usually appears a problem of nondeterministic choice of one action possible to apply in a given state. Therefore, usually there may be many solutions to a given planning problem consisting in bringing a complex object from the initial state to the final one using different plans. Meanwhile, in practical applications there often appears a situation that the automatically generated plan must be compatible with the plan proposed by the expert (*e.g.*, the treatment plan should be compatible with the plan proposed by human experts from a medical clinic). Hence, we inevitably need tools which may be used during a plan generation to solve the conflicts appearing between actions which may be performed at a given planning state. It also concerns making the decision about what state results from the action performed. That is why, in the paper, we propose a method which indicates the action to be performed in a given state or shows the state which is the result of the indicated action. This method uses a special classifier constructed on the basis of data sets and domain knowledge (see Section 1.3).

Problem of Synchronizing Plans. In planning the behavior of structurally complex objects consisting of parts being objects of lesser complexity, it is often not possible to plan effectively the behavior of a whole such object. That is why, in such cases the behavior of all parts is usually planned separately. However, such an approach to behavior planning for a complex object requires plan synchronization constructed for individual parts in such a way as not to make these plans contradicting one to another but be complement in order to plan the best behavior for the whole complex object. For example, treatment of a certain illness A, which is the result of illnesses B and C requires such a treatment planning of illnesses B and C so as not to make their treatments contradictory, but to make them to support and to complement one another during treatment of illness A. In the paper, *a planning synchronization method* for parts of a complex object is presented. It uses two classifiers constructed on the basis of data sets and domain knowledge (see Section 1.3). If we treat plans constructed for parts of a structured object as processes of some kind, then the method of synchronizing those plans is a method of synchronization of processes corresponding to the parts of a structured object. It should be emphasized, however, that the significant novelty of the method of synchronization of processes presented herein in relation to the ones known from literature (see, *e.g.*, [154, 155, 156, 157, 158, 159]) is the fact that the synchronization is carried out by using classifiers determined on the basis of data sets and domain knowledge.

Plan Adaptation Problem. After constructing a plan for a complex object, the execution of this plan may take place. However, the execution of the whole plan is not always possible in practice. It may happen that, during the plan execution such a state of complex object occurred that is not compatible with the state predicted by the plan. Then, the question arises whether the plan should still be executed or whether it should be reconstructed (updated).

If the current complex object state differs slightly from the state expected by the plan, then the execution of the current plan may perhaps be continued. If, however, the current state differs significantly from the state from the plan, then the current plan has to be reconstructed. It would seem that the easiest way to reconstruct the plan is construction of a new plan which commences at the current state of the complex object and ends at the final state of the old plan (a total reconstruction of the plan). However, in practical applications, a total reconstruction can be too costly in terms of computation or resources. Therefore, we need other methods which can effectively reconstruct the original plan in such a way as to realize it at least partially. Hence, in the paper, we propose a method of plan reconstruction called *a partial reconstruction*. It consists of constructing a short so-called *repair plan* which quickly brings the complex object to the so-called *return state* from the current plan. Next, on the basis of the repair plan, a reconstruction of the current plan is performed by replacing its fragment beginning with the current state and ending with the return state of the repair plan (see Section 1.3).

It is worth noticing that this issue is related to the domain of artificial intelligence called *the reasoning about changes* (see, *e.g.*, [160, 161]). Research works in this domain very often concern construction of a method of concluding about changes in satisfiability of concepts on a higher level of a certain concept hierarchy as a basis for discovery of plans aimed at restoration of the satisfiability of the desired concepts on a lower level of this hierarchy.

Problem of Similarity Relation Approximation. In building classifiers approximating complex spatio-temporal concepts, there may appear a need to estimate the similarity or the difference of two elements of a similar type such as complex objects, complex object states or plans generated for complex objects. This is an example of a classical case of *the problem of defining similarity relation* (or perhaps defining *dissimilarity relation* complementary to it) which is still one of the greatest challenges of data mining and knowledge discovery. The existing methods of defining similarity relations are based on building similarity functions on the basis of simple strategies of fusion of local similarities of compared elements. Optimization of the similarity formula established is performed by tuning both parameters of local similarities and their linking parameters (see, *e.g.*, [162, 163, 164, 165, 166, 167, 168, 169, 170, 171]). Frequently, however, experts from a given domain are not able to provide such a formula that would not raise their doubts and they limit themselves to the presentation of a set of examples of similarity function values, that is, a set of pairs of the compared elements labeled with degrees representing similarity function value. In this case, defining the similarity function requires its approximation with the help of a classifier, and at the same time such properties of compared elements should be defined that enable to approximate the similarity function. The main difficulty of the similarity function approximation is an appropriate choice of these properties. Meanwhile, according to the domain knowledge there are usually many various aspects of similarity between compared elements. For example, when comparing medical plans constructed for treatment of infants with a respiratory failure (see Appendix B), similarity of antibiotic therapies, similarity of applied mechanical ventilation methods, similarity of PDA closing and others should be taken into account. Each of these aspects should be considered in a specific way and presentation of formulas describing them can be extremely difficult for an expert. Frequently, an expert may only give examples of pairs of comparable elements together with their similarity in each of these aspects. Moreover, a fusion of different similarity aspects into a global similarity should also be performed in a way resulting from the domain knowledge. This way may be expressed, for example, using a concept ontology. In the paper, we propose *a method of similarity relation approximation* based on the usage of data sets and domain knowledge expressed, among other things, on the basis of a concept ontology (see Section 1.3).

1.3 Overview of the Results Achieved

As we mentioned before, the aim of this paper is to present a set of approximation methods of complex spatio-temporal concepts and approximate reasoning concerning these concepts, assuming that the information about concepts is given mainly in the form of a concept ontology.

The results described in the paper may be divided into the following groups:

1. methods for construction of classifiers stratifying a given concept,
2. general methodology of concept approximation with the usage of data sets and domain knowledge represented mainly in the form of a concept ontology,
3. methods for approximation of spatial concepts from an ontology,
4. methods for approximation of spatio-temporal concepts from an ontology defined for unstructured objects,
5. methods for approximation spatio-temporal concepts from an ontology defined for structured objects,
6. methods for behavioral pattern identification of complex objects in states of complex dynamical systems,

7. methods for automated planning of behavior of complex objects when the object states are represented by vague complex concepts requiring approximation,
8. implementation of all more crucial methods described in the paper as the RSES system extension.

In further subsections we briefly characterize the above groups of results.

At this point we present the publications on which the main results of our research have been partially based. The initial version of method for approximation of spatial concepts from an ontology was described in [172]. Methods for approximation of spatio-temporal concepts and methods for behavioral pattern identification were presented in [88, 173, 174, 175, 176, 177, 178]. Papers [173, 176, 177, 178] concern behaviors related to recognition of vehicle behavioral patterns or a group of vehicles on the road. The traffic simulator used to generate data for the needs of computer experiments was described in [179]. The paper [174] concerns medical applications related to recognition of high death risk pattern of infants suffering from respiratory failure, whereas papers [88, 175] concern both applications which were mentioned above. Finally, methods for automated planning of behavior of complex objects were described in [88, 180, 181].

Methods for Construction of Classifiers Stratifying Concepts. In practice, construction of classifiers often takes place on the basis of data sets containing uncertain and imprecise information (knowledge). That is why it is not often possible to construct a classifier which decisively classifies objects to the concept or its complement. This phenomenon occurs particularly when there is a need to classify objects not occurring in a learning set of objects, that is, those which are not used to construct the classifier.

One possible approach is to search for classifiers approximating probability distribution (see, *e.g.*, [6, 43]). However, in application, one may often require a less exact method based on classifying objects to different linguistic layers of the concept. This idea is inspired by papers of Professor Lotfi Zadeh (see, *e.g.*, [129, 130, 131]). In our approach, the discovered concept layers are used as patterns in searching for approximation of a more compound concept. In the paper, we present methods for construction of classifiers which, instead of stating clearly whether a tested object belongs to the concept or not, enable to obtain some membership degree of the tested object to the concept. In the paper, we define the concept of *a stratifying classifier* as a classifying algorithm stratifying concepts, that is, classifying objects to different concept layers (see Section 3). We propose two approaches to construction of these classifiers. One of them is *the expert approach* which is based on the defining, by an expert, an additional attribute in data which describes membership of the object to individual concept layers. Next, a classifier differentiating layers as decision classes is constructed. The second approach called *the automated approach* is based on the designing algorithms being the classifier extensions which enable to classify objects to concept layers on the basis of certain premises and experimental observations. In the paper, a new method of this type is proposed which is based on shortening of decision rules relatively to various coefficients of consistency.

General Methodology of Concept Approximation from Ontology. One of the main results presented in this paper is a methodology of approximating concepts from ontology. Generally, in order to approximate concepts a classical in machine learning [10] method of concept approximation is applied on the basis of positive and negative examples. It is based on the construction of a data table for each concept, known in rough set theory as a decision table (a special information system with a distinguished attribute called *decision* [16]) with rows (called objects) corresponding to positive and negative examples of the concept approximated and columns describing properties (features, attributes) of examples expressed by formulas in a considered language. The last column, called the decision column, is treated as a description of membership of individual examples to the concept approximated. For a table constructed in such a way, classifiers approximating a concept are built.

In such an approach, the main problem is the choice of examples of a given concept and properties of these examples.

The specificity of methodology of concept approximation proposed here in comparison with other methods (see, *e.g.*, [11, 52, 182]) is the usage of a domain knowledge expressed in the form of a concept ontology together with the rough set methods.

For concepts from the lowest level of an ontology hierarchy (the sensor level), not depending on the remaining concepts, we assume that so-called sensor attributes are also available which on the basis of given positive and negative examples, enable approximating these concepts by using classical methods of classifier construction.

However, the concept approximation methods, applied on a higher level of ontology consist in approximation of concepts using concepts from the lower ontology level. In this way, there are created hierarchical classifiers which use domain knowledge recorded in the form of ontology levels. In other words, patterns discovered for approximation of concepts on a given hierarchy level are used in construction of more compound patterns relevant for approximation of concepts on the next hierarchy level.

To approximate concepts from the higher ontology level, sensor attributes cannot be applied directly because the "semantical distance" of the higher level concepts from sensor attributes is too long and they are defined on different abstraction levels, i.e., searching for relevant features to approximate such concepts directly from sensory features becomes unfeasible (see the first problem from Section 1.2). For example, it is hardly believable that given only sensor attributes describing simple parameters of driving a vehicle (*e.g.*, location, speed, acceleration), one can approximate such a complex concept as *safe driving a vehicle*. Therefore, we propose a method, by means of which concepts from the higher ontology level exclusively be approximated by concepts from one level below. The proposed approach to concept approximation of a higher level is based on the assumption that the concept from the higher ontology level is semantically not too far from concepts lying on the lower level in the ontology. "Not too far" means that it may be expected that it is possible to approximate a concept from the higher ontology level with the help of lower ontology level concepts and patterns used for or derived from their construction, for which classifiers have already been built.

If we assume that approximation of concepts on the higher ontology level takes place using lower level concepts, then according to an established concept approximation methodology, positive and negative examples of the concept approximated are needed as well as their properties which serve the purpose of approximation. However, because of the semantical differences between concepts on different ontology levels, mentioned above, examples of lower ontology level concepts cannot be directly used to approximate a higher ontology level concept. For example, if the concept of a higher level concerns a group of vehicles (*e.g.*, *driving in a traffic jam*, *chase of one vehicle after another*, *overtaking*), whereas the lower level concepts concern single vehicles (*e.g.*, *accelerating*, *decelerating*, *changing lanes*), then the properties of a single vehicle (defined in order to approximate lower ontology level concepts) are usually insufficient to describe the properties of the whole group of vehicles. Difficulties with concept approximation on the higher ontology level using examples of the lower ontology level also appear when on the higher ontology level there are concepts concerning a time period different than that one related to the concepts on the lower ontology level. For example, a higher level concept may concern a time window, that is, a certain period of time (*e.g.*, *vehicle acceleration*, *vehicle deceleration*), whereas the lower level concepts may concern a certain instant, that is, a time point (*e.g.*, *a small vehicle speed*, *location of vehicle in the right lane*).

Hence, we present a method for construction of positive and negative examples of a concept of a higher ontology level consisting, in a general case, in arrangement (putting together) sets of examples of concepts of the lower ontology level. At the same time we define and represent such sets using patterns expressed in languages describing properties of examples of concepts of lower level in the ontology. These sets (represented by patterns) are arranged according to the so-called constraints resulting from the domain knowledge and determining which sets (patterns) may be arranged and which cannot be arranged for the construction of examples of higher level concepts. Thus, object structures on higher hierarchical levels come into being through linking (with the consideration of certain constraints) of objects from lower levels (and more precisely sets of these objects described by patterns). Such an approach enables a gradual modeling properties of more and more complex objects. Starting with elementary objects, objects being their sets or sequences of such objects, sets of sequences, etc. are gradually modeled. Different languages expressing properties of, *e.g.*, elementary objects, object sequences, or sets of sequences correspond to different model levels.

A crucial innovation feature of methods presented here is the fact that to define patterns describing examples of a lower ontology level, classifiers constructed for these concepts are used.

The example construction process for higher ontology level concepts on the basis of lower level concepts proceeds in the following way. Objects which are positive and negative examples of lower ontology level concepts are elements of a certain relational structure domain. Relations occurring in such a structure express relations between these objects and may be used to extract sets of objects of the lower ontology level. Each extracted set of objects is also a domain of a certain relational structure, in which relations are defined using information from a lower level. The process of extraction of relational structures is performed in order to approximate a higher ontology level concept with the help of lower ontology level concepts. Hence, to extract relational structures we necessarily need the information about membership of lower level objects to the concepts from this level. Such information may be available for any tested object based on the application of previously created classifiers for the lower ontology level concepts. Let us note that classifiers stratifying concepts are of a special importance here. The language in which we define formulas (patterns) to extract new relational structures using relational structures and lower ontology level concepts, is called the *language for extracting relational structures* (ERS-language).

For relational structures extracted in such a way, properties (attributes) may be defined which lead to an information system whose objects are extracted relational structures and the attributes are the properties of these structures (RS-information system). Relational structure properties may be defined using patterns which are formulas in a language specially constructed for this purpose, i.e., in *a language for definnig features of relational structures* (FRS-language). For example, some of the languages used to define the properties of extracted relational structures, presented in this paper, use elements of temporal logics with linear time, *e.g.*, *Linear Temporal Logic* (see, *e.g.*, [183, 184, 185]).

Objects of RS-information system are often inappropriate to make their properties relevant for the approximation of the higher ontology level concepts. It is due to the fact that there are too many such objects and their descriptions are too detailed. Hence, when applied to the higher ontology level concept approximation, the extension of the created classifier would be too low, that is, the classifier would classify too small number of tested objects. Apart from that, the problem of computational complexity would appear which means that because of a large number of objects in such information systems, the number of objects in a linking table, constructed in order to approximate concepts determined in a set of objects of a complex structure, would be too large to construct a classifier effectively (see below).

That is why a grouping (clustering) of such objects is applied which leads to obtaining more general objects, i.e., clusters of relational structures. This grouping may take place using a language chosen by an expert and called the *language for extracting clusters of relational structures* ($ECRS$-language). Within this language, a family of patterns may be selected to extract relevant clusters of relational structures from the initial information system.

For the clusters of relational structures obtained, an information system may be constructed whose objects are clusters defined by patterns from this family, and the attributes are the properties of these clusters. The properties of these clusters may be defined by patterns which are formulas of a language specially constructed for this purpose, i.e., *a language for defining features of clusters* *of relational structures* ($FCRS$-language). For example, some of the languages assigned to define the properties of relational structure clusters presented in this paper use elements of temporal logics with branching time, *e.g.*, *Branching Temporal Logic* (see, *e.g.*, [183, 184, 185]).

The information system with objects which are clusters of relational structures (CRS-information system) may already be used to approximate the concept of the higher ontology level. In order to do this, a new attribute is added to the system by the expert informs about membership of individual clusters to the concept approximated, and owing to that we obtain an approximation table of a higher ontology concept.

The method of construction of the approximation table of a higher ontology level concept may be generalized for concepts determined on a set of structured objects, that is, ones consisting of a set of parts (*e.g.*, *a group of vehicles on the road*, *a group of interacting illnesses*, *a robot team performing a task together*). This generalization means that CRS-information systems constructed for individual parts may be linked in order to obtain an approximation table of a higher ontology level concept determined for structured objects. Objects of this table are obtained through an arrangement (linking) of all possible objects of linked information systems. From the mathematical point of view this assumption is a Cartesian product of sets of objects of linked information systems. However, in terms of domain knowledge not all object links belonging to such a Cartesian product are possible (see [78, 84, 186, 187]). For example, if we approximate the concept of *safe overtaking*, it makes sense to arrange objects concerning only such vehicle pairs which are in the process of the overtaking maneuver.

For the reason mentioned above, that is, elimination of unrealistic complexes of objects, the so-called constraints are defined that are formulas built on the basis of arranged object features. The constraints determine which objects may be arranged in order to obtain an example of an object from a higher level and which may not. Additionally, we assume that to each arrangement allowed by the constraints, the expert adds a decision value informing whether a given arrangement belongs ore does not belong to the approximated concept of a higher level.

The table constructed in such a way serves the purpose of the approximation of a concept describing structured objects. However, in order to approximate a concept concerning structured objects, it is often necessary to construct not only all parts of the structured object but also features describing relations between parts. For example, *driving one vehicle after another*, apart from features describing the behavior of those two vehicles separately, features describing the location of these vehicles in relation to one another as well ought to be constructed. That is why in construction of a table of concept approximation for structured objects, there is constructed an additional CRS-information system whose attributes entirely describe the whole structured object in terms of relations between the parts of this object. In approximation of the object concerning structured objects, this system is arranged together with other CRS-information systems constructed for individual parts of the structured objects.

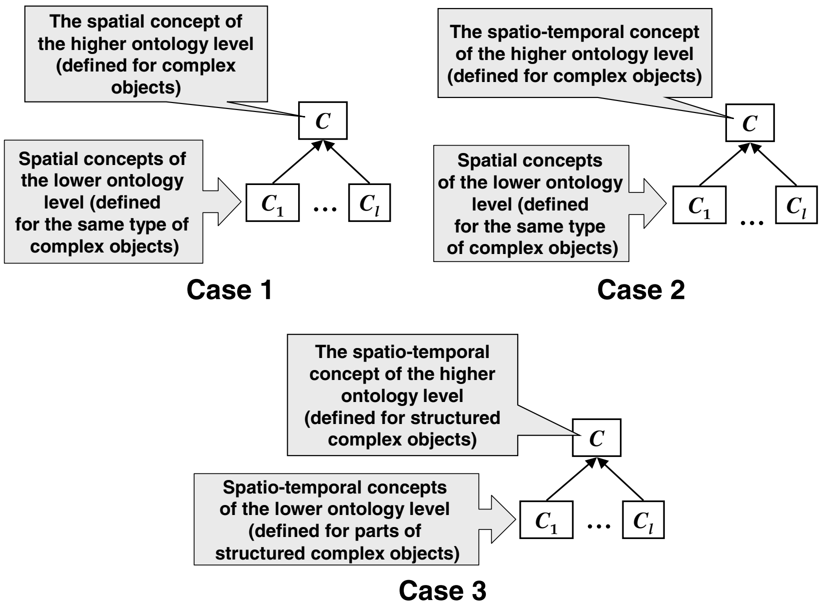

The spatial concept of the higher ontology level (defined for complex objects)
Spatial concepts of the lower ontology level (defined for the same type of complex objects)
C
C_1 ... C_l
Case 1
The spatio-temporal concept of the higher ontology level (defined for complex objects)
Spatial concepts of the lower ontology level (defined for the same type of complex objects)
Case 2
The spatio-temporal concept of the higher ontology level (defined for structured complex objects)
Spatio-temporal concepts of the lower ontology level (defined for parts of structured complex objects)
Case 3

Fig. 1. Three cases of complex concepts approximation in ontology

A fundamental problem in construction of an approximation table of a higher ontology level concept is, therefore, the choice of four appropriate languages used during its construction. The first language serves the purpose of defining patterns in a set of examples of a concept of lower ontology level which enable the relational structure extraction. The second one enables to define the properties of these structures. The third one makes possible to define relational structure clusters and, finally, the fourth one, the properties of these clusters. All these languages must be defined in such a way as to make the properties of the created relational structure clusters useful on a higher ontology level for approximation of the concept occurring there. Moreover, when the approximated concept concerns structured objects, each of the parts of this type of objects may require another four the languages similar to those already mentioned above.

Definitions of the above four languages depends on the semantical difference between concepts from both ontology levels. In the paper, the above methodology is applied in the three following cases in which the above four languages are defined in a completely different way:

1. The concept of the higher ontology level is a spatial concept (it does not require observing changes of objects over time) and it is defined on the set of the same objects (examples) as concepts of the lower ontology level, and at the same time the lower ontology level concepts are also spatial concepts (see Case 1 from Fig. 1).

2. The concept of the higher ontology level is a spatio-temporal concept (it requires observing object changes over time) and it is defined on a set of the same objects (examples) as the lower ontology level concepts. Moreover, the lower ontology level concepts are spatial concepts exclusively (see Case 2 from Fig. 1).
3. The concept of the higher ontology level is a spatio-temporal concept defined on a set of objects which are structured objects in relation to objects (examples) of the lower ontology level concepts, that is, the lower ontology level objects are parts of objects from the higher ontology level. Additionally, and at the same time the lower ontology level concepts are also spatio-temporal concepts (see Case 3 from Fig. 1).

Methods described in the next three subsections concern the above three cases. These methods also found application in construction of methods of behavioral pattern identification and in automated planning.

Methods of Approximation of Spatial Concepts. In the paper, the method of approximating concepts from ontology is proposed when a higher ontology level concept is a spatial concept (not requiring an observation of changes over time) and it is defined on a set of the same objects (examples) as the lower ontology level concepts; at the same time, the lower level concepts are also spatial concepts. An exemplary situation of this type is an approximation of the concept of *Safe overtaking* (concerning single vehicles on the road) using concepts such as *Safe distance from the opposite vehicle during overtaking*, *Possibility of going back to the right lane* and *Possibility of safe stopping before the crossroads.*

The concept approximation method described in this subsection is an example of the general methodology of approximating concepts from ontology described previously. That is why its specificity is the domain knowledge usage expressed in the form of a concept ontology and application of rough set methods, mainly in terms of application of classifier construction methods.

The basic terms used in the presented method is *pattern* and *production rule.* Patterns are descriptions of examples of concepts from an ontology and they are constructed by classifiers stratifying these concepts. A production rule is a decision rule which is constructed on two adjacent levels of ontology. In the *predecessor* of this rule there are patterns for the concepts from the lower level of the ontology whereas in the *successor*, there is a pattern for one concept from the higher level of the ontology (connected with concepts from the rule predecessor) where both patterns from the predecessor and the successor of the rule are chosen from patterns constructed earlier for concepts from both adjacent levels of the ontology. A rule constructed in such a way may serve as a simple classifier or an argument "for"/"against" the given concept, enabling classification of objects which match the patterns from the rule predecessor with the pattern from the rule successor. In the paper, there is proposed an algorithmic method of induction of production rules, consisting in an appropriate search for data tables with attributes describing the membership of training objects to particular layers of concepts (see Section 5.4). These tables are constructed using the so-called constraints between concepts thanks to which the information put in the tables only concerns those objects/examples which might be found there according to the production rule under construction.

Although a single production rule may be used as a classifier for the concept appearing in a rule successor, it is not a complete classifier yet, i.e., classifying all objects belonging to an approximated concept and not only those matching patterns of a rule predecessor. Therefore, in practice, production rules are grouped into the so-called *productions* (see Section 5.3), i.e., production rule collections, in a way that each production contains rules having patterns for the same concepts in a predecessor and the successor, but responding to their different layers. Such production is able to classify much more objects than a single production rule where these objects are classified into different layers of the concept occurring in a rule successor. Both productions and production rules themselves are only constructed for the two adjacent levels of ontology. Therefore, in order to use the whole ontology fully, there are constructed the so-called AR-schemes, i.e., *approximate reasoning schemes* (see, *e.g.*, [77, 89, 172, 188, 189, 190, 191, 192, 193, 194]) which are hierarchical compositions of production rules (see Section 5.7). The synthesis of an AR-scheme is carried out in a way that to a particular production from a lower hierarchical level of the AR-scheme under construction another production rule on a higher level may be attached, but only that one where one of the concepts for which the pattern occurring in the predecessor was constructed is the concept connected with the rule successor from the previous level. Additionally, it is required that the pattern occurring in a rule predecessor from the higher level is a subset of the pattern occurring in a rule successor from the lower level (in the sense of inclusion of object sets matching both patterns). To the two combined production rules other production rules can be attached (from above, from below or from the side) and in this way a multilevel structure is made which is a composition of many production rules. The AR-scheme constructed in such a way can be used as a hierarchical classifier whose entrance are predecessors of production rules from the lowest part of the AR-scheme hierarchy and the exit is the successor of a rule from the highest part of the AR-scheme hierarchy. That way, each AR-scheme is a classifier for a concept occurring in the rule successor from the highest part in the hierarchy of the scheme and, to be precise, for a concept for which a pattern occurring in the rule successor from the highest part in the hierarchy of the AR-scheme is determined.

However, similarly to the case of a single production rule, an AR-scheme is not a full classifier yet. That is why, in practice, for a particular concept there are many AR-schemes constructed which approximate different layers or concept regions.

In this paper, there are proposed two approaches for constructing AR-schemes (see Section 5.7). The first approach is based on memory with AR-schemes and consists in building many AR-schemes after determining production, which later on are stored and used for the classification of tested objects.

The second approach is based on a dynamic construction of AR-schemes. It is realized in a way that during classification of a given tested object, an appropriate AR-schemes for classifying this particular object is built on the basis of a given collection of productions ("lazy" classification).

In order to test the quality and effectiveness of classifier construction methods based on AR-schemes, experiments on data generated from the traffic simulator were performed (see Section 5.8). The experiments showed that classification quality obtained through classifiers based on AR-schemes is higher than classification quality obtained through traditional classifiers based on decision rules. Apart from that, the time spent on classifier construction based on AR-schemes is shorter than when constructing classical rule classifiers, their structure is less complicated than that of classical rule classifiers (a considerably smaller average number of decision rules), and their performance is much more stable because of the differences in data in samples supplied for learning (*e.g.*, to change the simulation scenario).

Methods of Approximation of Spatio-temporal Concepts. We also propose a method of approximating concepts from ontology when a higher ontology level concept is a spatio-temporal concept (it requires observing changes of complex objects over time) defined on a set of the same objects as the lower ontology level concepts; at the same time, the lower ontology level concepts are spatial concepts only. This case concerns a situation when during an observation of a single object in order to capture its behavior described by a higher ontology level concept, we have to observe it longer than it requires to capture behaviors described by lower ontology level concepts. For example, lower ontology level concepts may concern simple vehicle behaviors such as *small increase in speed*, *small decrease in speed* or *small move towards the left lane*. However, the higher ontology level concept may be a more complex concept as, *e.g.*, *acceleration in the right lane*. Let us notice that determining whether a vehicle accelerates in the right lane requires its observation for some time called *a time window*. On the other hand, determining whether a vehicle speed increases in the right lane requires only a registration of the speed of a vehicle in two neighboring instants (time points) only. That is why spatio-temporal concepts are more difficult to approximate than spatial concepts whose approximation does not require observing changes of objects over time.

Similarly to spatial concept approximation (see above), the method of concept approximation described in this subsection is an example of the general methodology of approximating concepts from ontology described earlier. Its specificity is, therefore, the domain knowledge usage expressed in the form of a concept ontology and rough set method application, mainly in terms of application of classifier construction methods. However, in this case more complex ontologies are used, and they contain both spatial and spatio-temporal concepts.

The starting point for the method proposed is a remark that spatio-temporal concept identification requires an observation of a complex object over a longer period of time called *a time window* (see Section 6.4). To describe complex object changes in the time window, the so-called *temporal patterns* (see Section 6.6) are used, which are defined as functions determined on a given time window. These patterns, being in fact formulas from a certain language, also characterize certain spatial properties of the complex object examined, observed in a given time window. They are constructed using lower ontology level concepts and that is why identification whether the object belongs to these patterns requires the application of classifiers constructed for concepts of the lower ontology level.

On a slightly higher abstraction level, the spatio-temporal concepts (also called *temporal concepts*) are directly used to describe complex object behaviors (see Section 6.5). Those concepts are defined by an expert in a natural language and they are usually formulated using questions about the current status of spatio-temporal objects, *e.g.*, *Does the vehicle examined accelerate in the right lane?*, *Does the vehicle maintain a constant speed during lane changing?* The method proposed here is based on approximating temporal concepts using temporal patterns with the help of classifiers. In order to do this a special decision table is constructed called *a temporal concept table* (see Section 6.9). The rows of this table represent the parameter vectors of lower level ontology concepts observed in a time window (and, more precisely, clusters of such parameter vectors). Columns of this table (apart from the last one) are determined using temporal patterns. However, the last column represents membership of an object, described by parameters (features, attributes) from a given row, to the approximated temporal concept.

Temporal concepts may be treated as nodes of a certain directed graph which is called a *behavioral graph.* Links (directed edges) in this graph are the temporal relations between temporal concepts meaning a temporal sequence of satisfying two temporal concepts one after another. These graphs are of a great significance in complex objects approximation for structured objects (see below).

Methods of Approximation of Spatio-temporal Concepts for Structured Objects. The method of spatio-temporal concept approximation presented in the previous subsection is extended to the case when higher ontology level concepts are defined on a set of objects which are structured objects in relation to objects (examples) of the lower ontology level concepts, that is, the lower ontology level objects are parts of objects from the higher ontology level. Moreover, lower ontology level concepts are also spatio-temporal concepts. This case concerns a situation when during a structured object observation, which serves the purpose of capturing its behavior described by a higher ontology level concept, we must observe this object longer than it is required to capture the behavior of a single part of the structured object described by lower ontology level concepts. For example, lower ontology level concepts may concern complex behaviors of a single vehicle such as *acceleration in the right lane*, *acceleration and changing lanes from right to left*, *decelerating in the left lane.* However, a higher ontology level concept may be an even more complex concept describing behavior of a structured object consisting of two vehicles (the overtaking and the overtaken one) over a certain period of time, for example, *the overtaking vehicle changes lanes from right to left, whereas the overtaken vehicle drives in the right lane.* Let us notice that the behavior described by this concept is a crucial fragment of the overtaking maneuver and determining whether the observed group of two vehicles behaved exactly that way, requires observing a sequence of behaviors of vehicles taking part in this maneuver for a certain period of time. They may be: *acceleration in the right lane*, *acceleration and changing lanes from right to left*, *maintaining a stable speed in the right lane.*

Analogously to the case of spatial and spatio-temporal concept approximation for unstructured objects, the method of concept approximation described in this subsection is an example of the general methodology of approximating concepts from ontology described previously. Hence, its specificity is also the domain knowledge usage expressed in the form of a concept ontology and rough set methods. However, in this case, ontologies may be extremely complex, containing concepts concerning unstructured objects, concepts concerning structured objects as well as concepts concerning relations between parts of structured objects.

The starting point for the proposed method is the remark that spatio-temporal concept identification concerning structured objects requires observing changes of these objects over a longer period of time (the so-called longer time windows) than in the case of complex objects which are parts of structured objects. Moreover, spatio-temporal concept identification concerning structured objects requires not only an observation of changes of all constituent parts of a given structured object individually, but also an observation of relations between these constituent parts and changes concerning these relations. Therefore, in order to identify spatio-temporal concepts concerning structured objects in behavioral graphs, we may observe paths of their constituent objects corresponding to constituent part behaviors in a given period. Apart from that paths in behavioral graphs describing relation changes between parts of structured objects should be observed. The properties of these paths may be defined using functions which we call *temporal patterns for temporal paths* (see Section 6.17). These patterns, being in fact formulas from a certain language, characterize spatio-temporal properties of the examined structured object in terms of its parts and constraints between these parts. On a slightly higher abstraction level, to describe behaviors of structured objects, the so-called *temporal concepts for structured objects* (see Section 6.20) are used, which are defined by an expert in a natural language and formulated usually with the help of questions about the current status of structured objects, *e.g.*, *Does one of the two observed vehicles approach the other driving behind it in the right lane?*, *Does one of the two observed vehicles change lanes from the right to the left one driving behind the second vehicle?*

The method of temporal concept approximation concerning structured objects, proposed here, is based on approximation of temporal concepts using temporal patterns for paths in behavioral graphs of parts of structured objects with the usage of temporal patterns for paths in behavioral graphs reflecting relation changes between the constituent parts. In order to do this a special decision table is constructed called *a temporal concept table of structured objects* (see Section 6.20). The rows of this table are obtained by arranging feature (attribute) value vectors of paths from behavioral graphs corresponding to parts of the structured objects observed in the data set (and, more precisely, value vectors of cluster features of such paths) and value vectors of path features from the behavioral graph reflecting relation changes between parts of the structured object (and, more precisely, value vectors of cluster features of such paths). From the mathematical point of view such an arrangement is a Cartesian product of linked feature vectors. However, in terms of domain knowledge not all links belonging to such a Cartesian product are possible and making sense (see [78, 84, 186, 187]).

According to the general methodology presented above, to eliminate such arrangements of feature vectors that are unreal or do not make sense, we define the so-called constraints which are formulas obtained on the basis of values occurring in the vectors arranged. The constraints determine which vectors may be arranged in order to obtain an example of a concept from a higher level and which may not. Additionally, we assume that to each feature vector arrangement, acceptable by constraints, the expert adds the decision value informing about the fact whether a given arrangement belongs to the approximated concept from the higher level.

Methods of Behavioral Pattern Identification. Similarly to the case of spatio-temporal concepts for unstructured complex objects, the spatio-temporal concepts defined for structured objects may also be treated as nodes of a certain directed graph which is called *a behavioral graph for a structured object* (see Section 6.22).

These graphs may be used to represent and identify the so-called behavioral patterns which are complex concepts concerning dynamic properties of complex structured objects expressed in a natural language depending on time and space. Examples of behavioral patterns may be: *overtaking on the road*, *driving in a traffic jam*, *behavior of a patient connected with a high life threat*. These types of concepts are much more difficult to approximate even than many temporal concepts.

In the paper, a new method of behavioral pattern identification is presented which is based on interpreting the behavioral graph of a structured object as a complex classifier enabling identification of a behavioral pattern described by this graph. This is possible based on the observation of the structured object behavior for a longer time and checking whether the behavior matches the chosen behavioral graph path. If this is so, then it is determined if the behavior matches the behavioral pattern represented by this graph, which enables a detection of specific behaviors of structured objects (see Section 6.23).

The effective application of the above behavioral pattern identification method encounters, however, two problems in practice. The first of them concerns extracting relevant context for the parts of structured objects (see the fourth problem from Section 1.2). To solve this problem *a sweeping method*, enabling a rapid structured object extraction, is proposed in this paper. This method works on the basis of simple heuristics called *sweeping algorithms around complex objects* which are constructed with the use of a domain knowledge supported by data sets (see Section 6.13).

The second problem appearing with behavioral pattern identification is the problem of fast elimination of such objects that certainly do not match a given behavioral pattern (see the fifth problem from Section 1.2). As one of the methods of solving this problem, we proposed the so-called method of fast elimination of specific behavioral patterns in relation to the analyzed structured objects. This method is based on the so-called *rules of fast elimination of behavioral patterns* which are determined from the data and on the basis of a domain knowledge (see Section 6.24). It leads to a great acceleration of behavioral pattern identification because such structured objects, whose behavior certainly does not match a given behavioral pattern, may be very quickly eliminated. For these objects it is not necessary to apply the method based on behavioral graphs which greatly accelerates the global perception.

In order to test the quality and effectiveness of classifier construction methods based on behavioral patterns, there have been performed experiments on data generated from the road simulator and medical data connected to detection of higher-death risk in infants suffering from the respiratory failure (see Section 6.25 and Section 6.26). The experiments showed that the algorithmic methods presented in this paper provide very good results in detecting behavioral patterns and may be useful with complex dynamical systems monitoring.

Methods of Automated Planning. Automated planning methods for unstructured complex objects were also worked out. These methods work on the basis of data sets and a domain knowledge represented by a concept ontology. A crucial novelty in the method proposed here, in comparison with the already existing ones, is the fact that performing actions according to plan depends on satisfying complex vague spatio-temporal conditions expressed in a natural language, which leads to the necessity of approximation of these conditions as complex concepts. Moreover, these conditions describe complex concept changes which should be reflected in the concept ontology.

Behavior of unstructured complex objects is modeled using the so-called *planning rules* being formulas of the type: *the state before performing an action* $\rightarrow$ *action* $\rightarrow$ *state* 1 *after performing an action* | ... | *state k after performing an action*, which are defined on the basis of data sets and a domain knowledge (see Section 7.4). Each rule includes the description of the complex object state before applying the rule (that is, before performing an action), expressed in a language of features proposed by an expert, the name of the action (one of the actions specified by the expert which may be performed at a particular state), and the description of sequences of states which a complex object may turn into after applying the action mentioned above. It means that the application of such a rule gives indeterministic effects, i.e., after performing the same action the system may turn into different states. All planning rules may be represented in a form of the so-called *planning graphs* whose nodes are state descriptions (occurring in predecessors and successors of planning rules) and action names occurring in planning rules (see Section 7.4). In the graphical interpretation, solving the problem of automated planning is based on finding a path in the planning graph from the initial state to an expected final state. It is worth noticing that the conditions for performing an action (object states) are described by vague spatio-temporal complex concepts which are expressed in the natural language and require an approximation.

For specific applications connected with the situation when it is expected that the proposed plan of a complex object behavior is to be strictly compatible with the determined experts' instructions (*e.g.*, the way of treatment in a specialist clinic is to be compatible with the treatment schemes used there), there has also been proposed an additional mechanism enabling to resolve the nondeterminism occurring in the application of planning rules. This mechanism is an additional classifier based on data sets and domain knowledge. Such classifiers suggest the action to be performed in a given state and show the state which is the result of the indicated action (see Section 7.7).

The automated planning method for unstructured objects has been generalized in the paper also in the case of planning of the behavior of structured objects (consisting of parts connected with one another by dependencies). The generalization is based on the fact that on the level of a structured object there is an additional planning graph defined where there are double-type nodes and directed edges between the nodes (see Section 7.11). The nodes of the first type describe vague features of states (meta-states) of the whole structured object, whereas the nodes of the second type concern complex actions (meta-actions) performed by the whole structured object (all its constituent parts) over a longer period of time (a time window). The edges between the nodes represent temporal dependencies between meta-states and meta-actions as well as meta-actions and meta-states. Similarly to the previous case of unstructured objects, planning of a structured object behavior is based on finding a path in a planning graph from the initial meta-state to the expected final meta-state; and, at the same time, each meta-action occurring in such a path must be planned separately on the level of each constituent part of the structured object. In other words, it should be planned what actions each part of a structured object must perform in order for the whole structured object to be able to perform the meta-action which has been planned. During the planning of a meta-action a synchronization mechanism (determining compatibility) of plans proposed for the part of a structured object is used, which works on the basis of a family of classifiers determined on the basis of data sets with a great support of domain knowledge. Apart from that, an additional classifier is applied (also based on a data set and the domain knowledge) which enables to determine whether the juxtaposition and execution of plans determined for the constituent parts, in fact, lead to the execution of the meta-action planned on the level of the whole structured object (see Section 7.13).

During the attempt to execute the plan constructed there often appears a need to reconstruct the plan which means that during the plan execution there may appear such a state of a complex object that is not compatible with the state suggested by the plan. A *total reconstruction* of the plan (building the whole plan from the beginning) may computationally be too costly. Therefore, we propose another plan reconstruction method called *a partial reconstruction*. It is based on constructing a short so-called *repair plan*, which rapidly brings the complex object to the so-called *return state* which appears in the current plan. Next, on the basis of the repair plan, a current plan reconstruction is performed through replacing its fragment beginning with the current state and ending with the return plan with the repair plan (see Section 7.9 and Section 7.17).

In construction and application of classifiers approximating complex spatio-temporal concepts, there may appear a need to construct, with a great support of the domain knowledge, a similarity relation of two elements of similar type, such as complex objects, complex object states, or plans generated for complex objects. Hence, in this paper we propose a new method of similarity relation approximation based on the use of data sets and a domain knowledge expressed mainly in the form of a concept ontology. We apply this method, among other things, to verify automated planning methods, that is, to compare the plan generated automatically with the plan suggested by experts from a given domain (see Section 7.18, Section 7.19 and Section 7.20).

In order to check the effectiveness of the automated planning methods proposed here, there were performed experiments concerning planning of treatment of infants suffering from the respiratory failure (see Section 7.21). Experimental results showed that the proposed method gives good results, also in the opinion of medical experts (compatible enough with the plans suggested by the experts), and may be applied in medical practice as a supporting tool for planning of the treatment of infants suffering from the respiratory failure.

Implementation and Data Sets. The result of the works conducted is also a programming system supporting the approximation of spatio-temporal complex concepts in the given concept ontology in the dialogue with the user. The system also includes an implementation of the algorithmic methods presented in this paper and is available on the web side of RSES system (see [15]).

Sections 5, 6 and 7, apart from the method description, contain the results of computing experiments conducted on real-life data sets, supported by domain knowledge. It is worth mentioning that the requirements regarding data sets which can be used for computing experiments with modeling spatio-temporal phenomena are much greater than the requirements of the data which are used for testing process of classical classifiers. Not only have the data to be representative of the decision making problem under consideration but also they have to be related to the domain knowledge available (usually cooperation with experts in a particular domain is essential). It is important that such data should fully and appropriately reflect complex spatio-temporal phenomena connected to the environment of the data collected.

The author of the paper acquired such data sets from two sources. The first source of data is the traffic simulator made by the author (see Appendix A). The simulator is a computing tool for generating data sets connected to the traffic on the street and at crossroads. During simulation each vehicle appearing on the simulation board behaves as an independently acting agent. On the basis of observation of the surroundings (other vehicles, its own location, weather conditions, etc.) this agent makes an independent decision what maneuvers it should make to achieve its aim which is to go safely across the simulation board and to leave the board using the outbound way given in advance. At any given moment of the simulation, all crucial vehicle parameters may be recorded, and thanks to this data sets for experiments can be obtained.

The second collection of data sets used in computer experiments was provided by Neonatal Intensive Care Unit, First Department of Pediatrics, Polish-American Institute of Pediatrics, Collegium Medicum, Jagiellonian University, Krakow, Poland. This data constitutes a detailed description of treatment of 300 infants, i.e., treatment results, diagnosis, operations, medication (see Section 6.26 and Appendix B).

1.4 Organization of the Paper

This paper is organized as follows. In Section 2 we briefly describe selected classical methods of classifier construction and concept approximation which are used in next subsections of the paper. These methods are based on rough set theory achievements and were described in the author's previous papers (see, *e.g.*, [14, 195, 196, 197, 198, 199, 200, 201, 202, 203]).

In Section 3 we describe methods of construction of a concept stratifying classifier.

The general methodology of approximating concepts with the use of data sets and a domain knowledge represented mainly in the form of a concept ontology is described in Section 4.

Methods of approximating spatial concepts from ontology are described in Section 5, whereas methods of approximating spatio-temporal concepts from ontology and methods of behavioral patterns identification are described in Section 6.

Methods of automated planning of complex object behavior when object states are represented with the help of complex objects requiring an approximation with the use of data sets and a domain knowledge are presented in Section 7.

Finally, in Section 8 we summarize the results and give directions for the future research.

The paper also contains two appendixes. The first appendix contains the description of the traffic simulator used to generate experimental data (see Appendix A). The second one describes medical issues connected with the infant respiratory failure (see Appendix B) concerning one of the data sets used for experiments.

2 Classical Classifiers

In general, the term *classify* means *arrange objects in a group or class based on shared characteristics* (see [1]). In this work, the term *classification* has a special meaning, *i.e.*, *classification* connotes any context in which some decision or forecast about object grouping is made on the basis of currently available knowledge or information (see, *e.g.*, [11, 204]).

A *classification algorithm* (*classifier*) is an algorithm which enables us to make a forecast repeatedly on the basis of accumulated knowledge in new situations (see, *e.g.*, [11]). Here we consider the classification provided by a classifying algorithm which is applied to a number of cases to classify objects unseen previously. Each new object is assigned to a class belonging to a predefined set of classes on the basis of observed values of suitably chosen attributes (features).

Many approaches have been proposed to construct classification algorithms. Among them we would like to mention classical and modern statistical techniques (see, *e.g.*, [11, 13]), neural networks (see, *e.g.*, [11, 13, 205]), decision trees (see, *e.g.*, [11, 206, 207, 208, 209, 210, 211, 212]), decision rules (see, *e.g.*, [10, 11, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223]) and inductive logic programming (see, *e.g.*, [11, 224]).

In this section, we consider methods implemented in our system RSES (*Rough Set Exploration System*) (see [14, 225, 226, 227, 228, 229, 230, 231]). RSES is a computer software system developed for the purpose of data analysis (the data is assumed to be in the form of an information system or a decision table, see Section 2.1). In construction of classifiers, which is the main step in the process od data analysis with RSES, elements of rough set theory are used. In this paper, we call these algorithms the *standard RSES methods* of classifier construction.

The majority of the standard RSES methods of classifier construction have been applied in more advanced methods of classifier construction, which will be presented in Sections 3, 5, 6, and 7. Therefore, in this section we only give a brief overview of that methods of classifier construction. These methods are based on *rough set theory* (see [16, 17, 232]). In the Section 2.1 we start with introduction of basic rough set terminology and notation, necessary for the rest of this paper (see Section 2.1).

The analysis of data in the RSES system proceeds according to the scheme presented in Fig. 2. First, the data for analysis has to be loaded/imported into the system. Next, in order to have a better chance for constructing (learning) a proper classifier, it is frequently advisable to transform the initial data set. Such transformation, usually referred to as *preprocessing*, may consist of several steps. RSES supports preprocessing methods which make it possible to manage

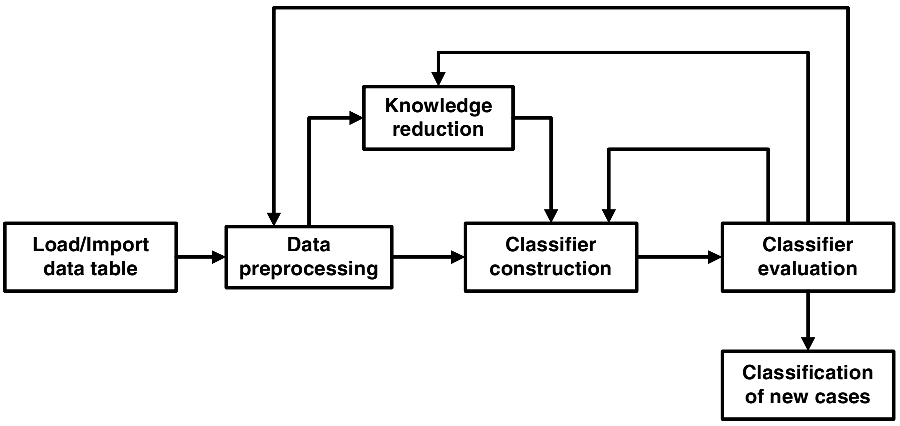


Fig. 2. The RSES data analysis process

missing parts in data, discretize numeric attributes, and create new attributes (see [14] and Section 2.2 for more details).

When the data is preprocessed, we can be interested in learning about its internal structure. By using classical rough set concepts such as reducts (see Section 2.1), dynamic reducts (see [14, 195, 196, 198, 201, 202, 203]), and positive region (see Section 2.1) one can discover dependencies that occur in our data set. Knowledge of reducts can lead to reduction of data by removing some of the redundant attributes.

Next, the classifier construction may be started. In the RSES system, these classifiers may be constructed using various methods (see [14] and sections 2.3, 2.4, 2.5, 2.6, 2.7 for more details).

A classifier is constructed on the basis of a training set consisting of labeled examples (objects with decisions). Such a classifier may further be used for evaluation on a test set or applied to new, unseen and unlabeled cases in order to determine the value of decision (classification) for them (see Section 2.9).

If the quality of the constructed classifier is insufficient, one may return to data preprocessing and/or knowledge reduction; another method of classifier construction may be applied as well.

2.1 Rough Set Basic Notions

In order to provide a clear description further in the paper and to avoid any misunderstandings, we bring here some essential definitions from rough set theory. We will frequently refer to the notions introduced in this section. Quite a comprehensive description of notions and concepts related to the classical rough set theory may be found in [189].

An *information system* (see [16, 17]) is a pair $\mathbf{A} = (U, A)$ where U is a non-empty, finite set called the *universe* of $\mathbf{A}$ and A is a non-empty, finite set of *attributes*, i.e., mappings $a : U \to V_a$, where V_a is called the *value set* of $a \in A$.

Elements of U are called *objects* and interpreted as, *e.g.*, cases, states, processes, patients, observations. Attributes are interpreted as features, variables, characteristic conditions.

We also consider a special case of information systems called decision tables. A *decision table* is an information system of the form $\mathbf{A} = (U, A, d)$ where $d \notin A$ is a distinguished attribute called the *decision*. The elements of A are called *condition attributes* or *conditions*.

One can interpret the decision attribute as a kind of partition of the universe of objects given by an expert, a decision-maker, an operator, a physician, etc. In machine learning decision tables are called training sets of examples (see [10]).

The cardinality of the image $d(U) = \{k : d(s) = k \text{ for some } s \in U\}$ is called the *rank of* d and is denoted by $r(d)$.

We assume that the set V_d of values of the decision d is equal to $\{v_d^1, ..., v_d^{r(d)}\}$.

Let us observe that the decision d determines a partition $CLASS_{\mathbf{A}}(d) = \{X_{\mathbf{A}}^1, \ldots, X_{\mathbf{A}}^{r(d)}\}$ of the universe U where $X_{\mathbf{A}}^k = \{x \in U : d(x) = v_d^k\}$ for $1 \le k \le r(d)$. $CLASS_{\mathbf{A}}(d)$ is called the *classification of objects of* $\mathbf{A}$ *determined* *by the decision d*. The set $X^i_{\mathbf{A}}$ is called the *i-th decision class of* $\mathbf{A}$. By $X_{\mathbf{A}}(u)$ we denote the decision class $\{x \in U : d(x) = d(u)\}$, for any $u \in U$.

Let $\mathbf{A} = (U, A)$ be an information system. For every set of attributes $B \subseteq A$, an equivalence relation, denoted by $IND_{\mathbf{A}}(B)$ and called the *B-indiscernibility relation*, is defined by

$$IND_{\mathbf{A}}(B) = \{(u, u') \in U \times U : \ \forall_{a \in B} \ a(u) = a(u')\}. \tag{1}$$

Objects u, u' being in the relation $IND_{\mathbf{A}}(B)$ are indiscernible by attributes from B.

By $[u]_{IND_{\mathbf{A}}(B)}$ we denote the equivalence class of the relation $IND_{\mathbf{A}}(B)$, such that u belongs to this class.

An attribute $a \in B \subseteq A$ is *dispensable* in B if $IND_{\mathbf{A}}(B) = IND_{\mathbf{A}}(B \setminus \{a\})$, otherwise a is *indispensable* in B. A set $B \subseteq A$ is *independent* in $\mathbf{A}$ if every attribute from B is indispensable in B, otherwise the set B is *dependent* in $\mathbf{A}$. A set $B \subseteq A$ is called *a reduct* in $\mathbf{A}$ if B is independent in $\mathbf{A}$ and $IND_{\mathbf{A}}(B) = IND_{\mathbf{A}}(A)$. The set of all reducts in $\mathbf{A}$ is denoted by $RED_{\mathbf{A}}(A)$. This is the classical notion of a reduct and it is sometimes referred to as *global reduct*.

Let $\mathbf{A} = (U, A)$ be an information system with n objects. By $M(\mathbf{A})$ (see [21]) we denote an $n \times n$ matrix (c_{ij}), called the *discernibility matrix* of $\mathbf{A}$, such that

$$c_{ij} = \{a \in A : a(x_i) \neq a(x_j)\} \ \text{ for } \ i, j = 1, \ldots, n\,. \tag{2}$$

A *discernibility function* $f_{\mathbf{A}}$ for an information system $\mathbf{A}$ is a Boolean function of m Boolean variables $\overline{a}_1, \ldots, \overline{a}_m$ corresponding to the attributes $a_1, \ldots, a_m$, respectively, and defined by

$$f_{\mathbf{A}}(\overline{a}_1, \ldots, \overline{a}_m) = \bigwedge\{\bigvee \overline{c}_{ij} : 1 \leq j < i \leq n \ \wedge \ c_{ij} \neq \emptyset\}, \tag{3}$$

where $\overline{c}_{ij} = \{\overline{a} : a \in c_{ij}\}$.

It can be shown (see [21]) that the set of all *prime implicants* of $f_{\mathbf{A}}$ determines the set of all *reducts* of A.

We present an exemplary deterministic algorithms for computation of the whole reduct set $RED_{\mathbf{A}}(A)$ (see, *e.g.*, [199]). This algorithm computes the discernibility matrix of $\mathbf{A}$ (see Algorithm 2.1).

The time cost of the reduct set computation using the algorithm presented above can be too high in the case the decision table consists of too many objects, attributes, or different values of attributes. The reason is that, in general, the size of the reduct set can be exponential with respect to the size of the decision table and the problem of the minimal reduct computation is NP-hard (see [21]). Therefore, we are often forced to apply approximation algorithms to obtain some knowledge about the reduct set. One way is to use approximation algorithms that need not give optimal solutions but require a short computing time. Among these algorithms are the following ones: Johnson's algorithm, covering algorithms, algorithms based on simulated annealing and Boltzmann machines, algorithms using neural networks and algorithms based on genetic algorithms (see, *e.g.*, [196, 198, 199] for more details).

Algorithm 2.1. Reduct set computation

```
   Input: Information system $\mathbf{A} = (U, A)$
   Output: Set $RED_{\mathbf{A}}(A)$ of all reducts of $\mathbf{A}$
 1 begin
 2     Compute indiscernibility matrix $M(\mathbf{A})$
 3     Reduce $M(\mathbf{A})$ using absorbtion laws
       // Let $C_1, ..., C_d$ are non-empty fields of reduced $M(\mathbf{A})$
 4     Build a familie of sets $R_0, R_1, ..., R_d$ in the following way:
 5     begin
 6         $R_0 = \emptyset$
 7         for $i = 1$ to $d$ do
 8             $R_i = S_i \cup T_i$ where $S_i = \{R \in R_{i-1} : R \cap C_i \neq \emptyset\}$
 9               and $T_i = (R \cup \{a\})_{a \in C_i, R \in R_{i-1} : R \cap C_i = \emptyset}$
10         end
11     end
12     Remove dispensable attributes from each element of family $R_d$
13     Remove redundant elements from $R_d$
14     $RED_{\mathbf{A}}(A) = R_d$
15 end
```

If $\mathbf{A} = (U, A)$ is an information system, $B \subseteq A$ is a set of attributes and $X \subseteq U$ is a set of objects (usually called a *concept*), then the sets $\{u \in U : [u]_{IND_{\mathbf{A}}(B)} \subseteq X\}$ and $\{u \in U : [u]_{IND_{\mathbf{A}}(B)} \cap X \neq \emptyset\}$ are called the *B-lower* and the *B-upper approximations* of X in $\mathbf{A}$, and they are denoted by $\underline{B}X$ and $\overline{B}X$, respectively.

The set $BN_B(X) = \overline{B}X - \underline{B}X$ is called the *B-boundary* of X (*boundary region*, for short). When $B = A$, we also write $BN_{\mathbf{A}}(X)$ instead of $BN_A(X)$.

Sets which are unions of some classes of the indiscernibility relation $IND_{\mathbf{A}}(B)$ are called *definable* by B (or *B-definable* in short). A set X is, thus, B-definable iff $\overline{B}X = \underline{B}X$. Some subsets (categories) of objects in an information system cannot be exactly expressed in terms of the available attributes but they can be defined roughly.

The set $\underline{B}X$ is the set of all elements of U which can be classified with certainty as elements of X, given a knowledge about these elements in the form of values of attributes from B; the set $BN_B(X)$ is the set of elements of U which one can classify neither to X nor to $-X$ having a knowledge about objects represented by B.

If the boundary region of $X \subseteq U$ is the empty set, i.e., $BN_B(X) = \emptyset$, then the set X is called *crisp* (*exact*) with respect to B; in the opposite case, i.e., if $BN_B(X) \neq \emptyset$, the set X is referred to as *rough* (*inexact*) with respect to B (see, *e.g.*, [17]).

If $X_1, \ldots, X_{r(d)}$ are decision classes of $\mathbf{A}$, then the set $\underline{B}X_1 \cup \cdots \cup \underline{B}X_{r(d)}$ is called *the B-positive region of* $\mathbf{A}$ and denoted by $POS_B(d)$.

If $\mathbf{A} = (U, A, d)$ is a decision table and $B \subseteq A$, then we define a function $\partial_B : U \to \mathbf{P}(V_d)$, called *the B-generalized decision of* $\mathbf{A}$, by

$$\partial_B(x) = \{v \in V_d : \exists x' \in U\, (x' IND_{\mathbf{A}}(B) x \text{ and } d(x) = v)\}\ . \tag{4}$$

The A-generalized decision ∂_A of $\mathbf{A}$ is called the *generalized decision* of $\mathbf{A}$.

A decision table $\mathbf{A}$ is called *consistent (deterministic)* if $card(\partial_A(x)) = 1$ for any $x \in U$, otherwise $\mathbf{A}$ is *inconsistent (non-deterministic).* Non-deterministic information systems were introduced by Witold Lipski (see [233]), while deterministic information systems independently by Zdzisław Pawlak [234] (see, also, [235, 236]). It is easy to see that a decision table $\mathbf{A}$ is consistent iff $POS_A(d) = U$. Moreover, if $\partial_B = \partial_{B'}$, then $POS_B(d) = POS_{B'}(d)$ for any pair of non-empty sets $B, B' \subseteq A$.

A subset B of the set A of attributes of a decision table $\mathbf{A} = (U, A, d)$ is *a relative reduct of* $\mathbf{A}$ iff B is a minimal set with respect to the following property: $\partial_B = \partial_A$. The set of all relative reducts of $\mathbf{A}$ is denoted by $RED(\mathbf{A}, d)$.

Let $\mathbf{A} = (U, A, d)$ be a consistent decision table and let $M(\mathbf{A}) = (c_{ij})$ be its discernibility matrix. We construct a new matrix $M'(\mathbf{A}) = (c'_{ij})$ assuming $c'_{ij} = \emptyset$ if $d(x_i) = d(x_j)$, and $c'_{ij} = c_{ij} - \{d\}$ otherwise. The matrix $M'(\mathbf{A})$ is called *the relative discernibility matrix of* $\mathbf{A}$. Now, one can construct the *relative discernibility function* $f_{M'(A)}$ of $M'(\mathbf{A})$ in the same way as the discernibility function.

It can be shown (see [21]) that the set of all *prime implicants* of $f_{M'(A)}$ determines the set of all *relative reducts* of $\mathbf{A}$.

Another important type of reducts are local reducts. *A local reduct* $r(x_i) \subseteq A$ (or a *reduct relative to decision and object* $x_i \in U$ where x_i is called a *base object*) is a subset of A such that:

1. $\forall_{x_j \in U}\ d(x_i) \neq d(x_j) \Longrightarrow \exists_{a_k \in r(x_i)}\ a_k(x_i) \neq a_k(x_j)$,
2. $r(x_i)$ is minimal with respect to inclusion.

If $\mathbf{A} = (U, A, d)$ is a decision table, then any system $\mathbf{B} = (U', A, d)$ such that $U' \subseteq U$ is called a *subtable* of $\mathbf{A}$.

A template of $\mathbf{A}$ is a formula $\bigwedge(a_i = v_i)$ where $a_i \in A$ and $v_i \in V_{a_i}$. A generalized template is a formula of the form $\bigwedge(a_i \in T_i)$ where $T_i \subset V_{a_i}$. An object *satisfies* (matches) a template if for every attribute a_i occurring in the template, the value of this attribute at a considered object is equal to v_i (belongs to T_i in the case of the generalized template). The template splits the original information system in the two distinct subtables containing objects that satisfy and do not satisfy the template, respectively.

It is worth mentioning that the notion of a template can be treated as a particular case of a more general notion, viz., that of *a pattern* (see Section 4.9).

2.2 Discretization

Suppose we have a decision table $\mathbf{A} = (U, A, d)$ where $card(V_a)$ is high for some $a \in A$. Then, there is a very low chance that a new object is recognized by rules generated directly from this table because the attribute value vector of a new object will not match any of these rules. Therefore, for decision tables with real (numerical) value attributes, some discretization strategies are built in order to obtain a higher quality of classification. This problem was intensively studied (see, *e.g.*, [199, 237, 238] for more details).

The process of discretization is usually realized in two following steps (see, *e.g.*, [14, 199, 237, 238]). First, the algorithm generates a set of cuts. By a *cut* for an attribute $a_i \in A$ such that V_{a_i} is an ordered set we denote a value $c \in V_{a_i}$. The cuts can be then used to transform the decision table. As a result we obtain a decision table with the same set of attributes but the attributes have different values. Instead of $a(x) = v$ for an attribute $a \in A$ and an object $x \in U$, we rather get $a(x) \in [c_1, c_2]$ where c_1 and c_2 are cuts generated for attribute a by a discretization algorithm. The cuts are generated in a way that the resulting intervals contain possibly most uniform sets of objects w.r.t decision.

The discretization method available in RSES has two versions (see, *e.g.*, [14, 199, 238]) that are usually called *global* and *local*. Both methods belong to a bottom-up approaches which add cuts for a given attribute one-by-one in subsequent iterations of algorithm. The difference between these two methods lies in the way in which the candidate for a new cut is evaluated. In the global method, we evaluate all objects in the data table at every step. In the local method, we only consider a part of objects that are related to the candidate cut, i.e., which have the value of the attribute considered currently in the same range as the cut candidate. Naturally, the second (local) method is faster as less objects have to be examined at every step. In general, the local method produces more cuts. The local method is also capable of dealing with nominal (symbolic) attributes. Grouping (quantization) of a nominal attribute domain with use of the local method always results in two subsets of attribute values (see, *e.g.*, [14, 199, 238] for more details).

2.3 Decision Rules

Let $\mathbf{A} = (U, A, d)$ be a decision table and let $V = \bigcup\{V_a : a \in A\} \cup V_d$. Atomic formulas over $B \subseteq A \cup \{d\}$ and V are expressions of the form $a = v$, called *descriptors* over B and V, where $a \in B$ and $v \in V_a$. The set $\mathbf{F}(B, V)$ of formulas over B and V is the least set containing all atomic formulas over B, V and closed with respect to the classical propositional connectives $\vee$ (disjunction), $\wedge$ (conjunction), and $\neg$ (negation).

Let $\varphi \in \mathbf{F}(B, V)$. Then, by $|\varphi|_{\mathbf{A}}$ we denote the meaning of φ in the decision table $\mathbf{A}$, i.e., the set of all objects of U with the property φ, defined inductively by

1. if φ is of the form $a = v$, then $|\varphi|_{\mathbf{A}} = \{x \in U : a(x) = v\}$,
2. $|\varphi \wedge \varphi'|_{\mathbf{A}} = |\varphi|_{\mathbf{A}} \cap |\varphi'|_{\mathbf{A}}$,
3. $|\varphi \vee \varphi'|_{\mathbf{A}} = |\varphi|_{\mathbf{A}} \cup |\varphi'|_{\mathbf{A}}$,
4. $|\neg\varphi|_{\mathbf{A}} = U - |\varphi\}_{\mathbf{A}}$.

The set $\mathbf{F}(A,\, V)$ is called the set of *conditional formulas of* $\mathbf{A}$ and is denoted by $\mathbf{C}(A,\, V)$.

Any formula of the form $(a_1 = v_1) \wedge ... \wedge (a_l = v_l)$ where $v_i \in V_{a_i}$ (for $i = 1, ..., l$) and $P = \{a_1, ..., a_l\} \subseteq A$ is called a *P-basic* formula of $\mathbf{A}$.

If φ is a *P-basic* formula of $\mathbf{A}$ and $Q \subseteq P$, then by φ/Q we mean the Q-basic formula obtained from the formula φ by removing from φ all its elementary subformulas $(a = v_a)$ such that $a \in P \setminus Q$.

A *decision rule* for $\mathbf{A}$ is any expression of the form $\varphi \Rightarrow d = v$ where $\varphi \in \mathbf{C}(A,\, V)$, $v \in V_d$, and $|\varphi|_{\mathbf{A}} \neq \emptyset$. Formulas φ and $d = v$ are referred to as the *predecessor* (*premise* of the rule) and the *successor* of the decision rule $\varphi \Rightarrow d = v$ respectively.

If r is a decision rule in $\mathbf{A}$, then by $Pred(r)$ we denote the predecessor of r and by $Succ(r)$ we denote the successor of r .

An object $u \in U$ is *matched* by a decision rule $\varphi \Rightarrow d = v_d^k$ (where $1 \leq k \leq r(d)$) iff $u \in |\varphi|_{\mathbf{A}}$. If u is matched by $\varphi \Rightarrow d = v_d^k$, then we say that the rule is classifying u to the decision class X_k.

The number of objects matched by a decision rule $\varphi \Rightarrow d = v$, denoted by $Match_{\mathbf{A}}(\varphi \Rightarrow d = v)$, is equal to $card(|\varphi|_{\mathbf{A}})$.

The number $Supp_{\mathbf{A}}(\varphi \Rightarrow d = v) = card(|\varphi|_{\mathbf{A}} \cap |d = v|_{\mathbf{A}})$ is called *the number of objects supporting the decision rule* $\varphi \Rightarrow d = v$.

A decision rule $\varphi \Rightarrow d = v$ for $\mathbf{A}$ is *true* in $\mathbf{A}$, symbolically $\varphi \Rightarrow_{\mathbf{A}} d = v$, iff $|\varphi|_{\mathbf{A}} \subseteq |d = v|_{\mathbf{A}}$. If the decision rule $\varphi \Rightarrow d = v$ is true in $\mathbf{A}$, we say that the decision rule is *consistent* in $\mathbf{A}$, otherwise $\varphi \Rightarrow d = v$ is *inconsistent* or *approximate* in $\mathbf{A}$.

If r is a decision rule in $\mathbf{A}$, then the number $\mu_{\mathbf{A}}(r) = \frac{Supp_{\mathbf{A}}(r)}{Match_{\mathbf{A}}(r)}$ is called *the coefficient of consistency* of the rule r. The coefficient $\mu_{\mathbf{A}}(r)$ may be understood as the degree of consistency of the decision rule r. It is easy to see that a decision rule r for $\mathbf{A}$ is consistent iff $\mu_{\mathbf{A}}(r) = 1$.

The coefficient of consistency of r can be also treated as the degree of inclusion of $|Pred(r)|_{\mathbf{A}}$ in $|Succ(r)|_{\mathbf{A}}$ (see, *e.g.*, [239]).

If $\varphi \Rightarrow d = v$ is a decision rule for $\mathbf{A}$ and φ is P-basic formula of $\mathbf{A}$ (where $P \subseteq A$), then the decision rule $\varphi \Rightarrow d = v$ is called a *P-basic decision rule for* $\mathbf{A}$, or a *basic decision rule* in short.

Let $\varphi \Rightarrow d = v$ be a P-basic decision rule of $\mathbf{A}$ (where $P \subseteq A$) and let $a \in P$. We will say that the attribute a is *dispensable* in the rule $\varphi \Rightarrow d = v$ iff $|\varphi \Rightarrow d = v|_{\mathbf{A}} = U$ implies $|\varphi/(P \setminus \{a\}) \Rightarrow d = v|_{\mathbf{A}} = U$, otherwise attribute a is *indispensable* in the rule $\varphi \Rightarrow d = v$. If all attributes $a \in P$ are indispensable in the rule $\varphi \Rightarrow d = v$, then $\varphi \Rightarrow d = v$ will be called *independent* in $\mathbf{A}$.

The subset of attributes $R \subseteq P$ will be called a *reduct* of P-basic decision rule $\varphi \Rightarrow d = v$, if $\varphi/R \Rightarrow d = v$ is independent in $\mathbf{A}$ and $|\varphi \Rightarrow d = v|_{\mathbf{A}} = U$ implies $|\varphi/R \Rightarrow d = v|_{\mathbf{A}} = U$. If R is a reduct of the P-basic decision rule $\varphi \Rightarrow d = v$, then $\varphi/R \Rightarrow d = v$ is said to be *reduced.* If R is a reduct of the A-basic decision rule $\varphi \Rightarrow d = v$, then $\varphi/R \Rightarrow d = v$ is said to be *an optimal basic decision rule of* $\mathbf{A}$ (*a basic decision rule with minimal number of descriptors*). The set of all optimal basic decision rules of $\mathbf{A}$ is denoted by $RUL(\mathbf{A})$.

2.4 Two Methods for Decision Rule Synthesis

Classifiers based on a set of decision rules are the most elaborated methods in RSES. Several methods for calculation of the decision rule sets are implemented. Also, various methods for transforming and utilizing rule sets are available. However, in our computer experiments we usually use two methods for decision rules synthesis. We would like to mention those methods here.

The first method returns all basic decision rules with minimal number of descriptors (see, *e.g.*, [196, 198, 199, 240]). Therefore, this method is often called *an exhaustive method*. From the practical point of view, the method consists in applying an algorithm computing all reducts (see Algorithm 2.1) for each object individually, which results in obtaining decision rules with a minimal number of descriptors in relation to individual objects (see, *e.g.*, [196, 198, 199]).

The second method for basic decision rule synthesis, is the covering algorithm called LEM2 (see, *e.g.*, [216, 222, 223]). In LEM2, a separate-and-conquer technique is paired with rough set notions such as upper and lower approximations. This method tends to produce less rules than algorithms based on the exhaustive local reduct calculation (as in the previous method) and seems to be faster. On the downside, the LEM2 method sometimes returns too few valuable and meaningful rules (see also Section 2.10).

2.5 Operations on Rule Sets

In general, the methods used by RSES to generate rules may produce quite a bunch of them. Naturally, some of the rules may be marginal, erroneous or redundant. In order to provide a better control over the rule-based classifiers some simple techniques for transforming rule sets should be used. The simplest way to alter a set of decision rules is by filtering them. It is possible to eliminate from the rule set these rules that have insufficient support on training sample, or those that point at a decision class other than the desired one. More advanced operations on rule sets are *shortening* and *generalization*.

Rule shortening is a method that attempts to eliminate descriptors from the premise of the rule. The resulting rule is shorter, more general (applicable to more training objects) but it may lose some of its precision. The shortened rule may be less precise, i.e., it may give wrong answers (decisions) for some of the matching training objects.

We present an exemplary method of approximate rules computation (see, *e.g.*, [196, 198, 199]) that we use in our experiments. We begin with an algorithm for synthesis of optimal decision rules from a given decision table (see Section 2.4). Next, we compute approximate rules from the optimal decision rules already calculated. Our method is based on the notion of consistency of a decision rule (see Section 2.1). The original optimal rule is reduced to an approximate rule with the coefficient of consistency exceeding a fixed threshold.

Let $\mathbf{A} = (U, A, d)$ be a decision table and $r_0 \in RUL(\mathbf{A})$. The approximate rule (based on rule r_0) is computed using the Algorithm 2.2.

Algorithm 2.2. Approximate rule synthesis (by descriptor dropping)

```
Input:
  1. decision table $\mathbf{A} = (U, A, d)$
  2. decision rule $r_0 \in RUL(\mathbf{A})$
  3. threshold of consistency $\mu_0$ (e.g., $\mu_0 = 0.9$)
Output: the approximate rule $r_{app}$ (based on rule $r_0$)
1  begin
2      Calculate the coefficient of consistency $\mu_{\mathbf{A}}(r_0)$
3      if $\mu_{\mathbf{A}}(r_0) < \mu_0$ then
4          STOP // In this case no approximate rule
5      end
6      $\mu_{max} = \mu_{\mathbf{A}}(r_0)$ and $r_{app} = r_0$
7      while $\mu_{max} > \mu_0$ do
8          $\mu_{max} = 0$
9          for $i = 1$ to the number of descriptors from $Pred(r_{app})$ do
10             $r = r_{app}$
11             Remove i-th descriptor from $Pred(r)$
12             Calculate the coefficient of consistency $\mu_{\mathbf{A}}(r)$ and $\mu = \mu_{\mathbf{A}}(r)$
13             if $\mu > \mu_{max}$ then
14                 $\mu_{max} = \mu$ and $i_{max} = i$
15             end
16         end
17         if $\mu_{max} > \mu_0$ then
18             Remove $i_{max}$ -th conditional descriptor from $r_{app}$
19         end
20     end
21     return $r_{app}$
22 end
```

It is easy to see that the time and space complexity of Algorithm 2.2 are of order $O(l^2 \cdot m \cdot n)$ and $O(C)$, respectively (where l is the number of conditional descriptors in the original optimal decision rule r_0 and C is a constant).

The approximate rules, generated by the above method, can help to extract interesting laws from the decision table. By applying approximate rules instead of optimal rules one can slightly decrease the quality of classification of objects from the training set but we expect, in return, to receive more general rules with a higher quality of classification of new objects (see [196]).

On the other hand, generalization of rules is a process which consists in replacement of the descriptors having a single attribute value in rule predecessors with more general descriptors. In the RSES system there is an algorithm available which instead of simple descriptors of type $a(x) = v$, where $a \in A$, $v \in V_a$ and $x \in U$ tries to use the so-called generalized descriptors of the form $a(x) \in V$ where $V \subset V_a$ (see, *e.g.*, [14]). In addition, such a replacement is performed only when the coefficient of consistency of the new rule is not smaller than the established threshold. Let us notice that such an operation is crucial in terms of enlargement of the extension of decision rules for the generalized decision rules are able to classify a greater number of tested objects.

It is worth mentioning that the application of the method of generalizing rules described above only makes sense for tables with attributes having a small number values. Such attributes are usually attributes with symbolic values. On the other hand a usage of this method for tables with numerical attributes requires a previous discretization of values of these attributes.

2.6 Negotiations Among Rules

Suppose we have a set of decision rules. When we attempt to classify an object from test sample with use of a rule set generated, it may happen that various rules suggest different decision values. In such conflict situations, we need a strategy to resolve controversy and reach a final result (decision). This problem was intensively studied (see, *e.g.*, [198, 199]). In its current version, RSES provides a conflict resolution strategy based on voting among rules. In this method, each rule that matches the object under consideration casts a vote in favor of the decision value it points at. Votes are summed up and the decision is chosen that has got majority of votes. This simple method may be extended by assigning weights to rules. Each rule, then votes with its weight and the decision that has the highest total of weighted votes is the final one. In RSES, this method is known as a *standard voting* and is based on a *basic strength* (weight) of decision rules (see Section 2.8). Of course, there are many other methods that can be used to resolve conflicts between decision rules (see, *e.g.*, [196, 198, 199, 216, 217, 241]).

2.7 Decomposition Trees

In the case of the decision tables larger, the computation of decision rules for these tables can be extremely difficult or even impossible.

This problem arises from a relatively high computational complexity of rule computing algorithms. Unfortunately, it frequently concerns covering algorithms such as, *e.g.*, LEM2 as well (see Section 2.4). One of the solutions to this problem is the so-called *decomposition*. Decomposition consists in partitioning the entrance data table into parts (subtables) in such a way as to be able to calculate decision rules for these parts using standard methods. Naturally, a method is also necessary which would aggregate the obtained rule sets in order to build a general classifier.

In this paper, we present a decomposition method based on a decomposition tree (see [165, 226, 242]) which may be constructed according to Algorithm 2.3.

This algorithm creates the decomposition tree in steps where each step leads to construction of the next level of the tree. At a given step of the algorithm execution, a binary partition of the decision table takes place using the best template (see Section 2.1) found for the table being partitioned. In this way, with each tree node (leaf), there is connected a template partitioning the subtable in this node into objects matching and not matching the template. This

Algorithm 2.3. Decomposition tree synthesis

```
Input: decision table $\mathbf{A} = (U, A, d)$
Output: the decomposition tree for the decision table $\mathbf{A}$
begin
    Find the best template $T$ in $\mathbf{A}$ (see Section 2.1)
    Divide $\mathbf{A}$ in two subtables: $\mathbf{A}_1$ containing all objects satisfying $\mathbf{T}$ and
    $\mathbf{A}_2 = \mathbf{A} - \mathbf{A}_1$
    if obtained subtables are of acceptable size in the sense of rough set
    methods then
        STOP // The decomposition is finished
    end
    repeat lines 2-7 for all "too large" subtables
end
```

template and its contradiction are transferred as templates describing subtables to the next step of decomposition. Decomposition finishes when the subtables obtained are so small that the decision rules can be calculated for them using standard methods. After determining the decomposition tree, decision rule sets are calculated for all the leaves of this tree and, more precisely, for the subtables occurring in single leaves.

The tree and the rules calculated for training sample can be used in classification of unseen cases. Suppose we have a binary decomposition tree. Let u be a new object, $\mathbf{A}(T)$ be a subtable containing all objects matching a template T, and $\mathbf{A}(\neg T)$ be a subtable containing all objects not matching a template T. We classify object u starting from the root of the tree using Algorithm 2.4.

This algorithm works in such a way that such a leaf of a decomposition tree is sought first that the tested object matches the template describing the objects of

Algorithm 2.4. Classification by decomposition tree

```
1  begin
2      if $u$ matches template $T$ found for $\mathbf{A}$ then
3          go to subtree related to $\mathbf{A}(T)$
4      else
5          go to subtree related to $\mathbf{A}(\neg T)$
6      end
7      if $u$ is at the leaf of the tree then
8          go to line 12
9      else
10         repeat lines 2-11 substituting $\mathbf{A}(T)$ (or $\mathbf{A}(\neg T)$) for $\mathbf{A}$
11     end
12     Classify $u$ using decision rules for subtable attached to the leaf
13 end
```

that leaf. Next, the object is classified with the help of decision rules calculated for the leaf that was found.

The type of the decomposition method depends on the method of determining the best template. For instance, if decomposition is needed only because it is impossible to compute rules for a given decision table, then the best template for this table is the template which divides a given table into two equal parts. If, however, we are concerned with the table partition that is most compatible with the partition introduced by decision classes, then the measure of the template quality may be, for example, the number of pairs of objects from different decision classes, differentiated with the help of the partition introduced by a given template. Surely, the best template in this case is a template with the largest number of differentiated pairs.

The patterns determined may have different forms (see, *e.g.*, [165] for more details). In the simplest case, for a symbolic attribute, the best template might be of the forms $a(x) = v$ or $a(x) \neq v$ where $a \in A$, $v \in V_a$, and $x \in U$, whereas for a numerical attribute, the templates might be $a(x) > v$, $a(x) < v$, $a(x) \leq v$, or $a(x) \geq v$ where $a \in A$, $v \in V_a$, and $x \in U$.

The classifier presented in this section uses a binary decision tree, however, it should not be mistaken for C4.5 or ID3 (see, *e.g.*, [210, 243]) because, as we said before, rough set methods have been used in leaves of the decomposition tree in construction of the classifying algorithm.

2.8 Concept Approximation and Classifiers

Definability of concepts is a term well-known in classical logic (see, *e.g.*, [5, 244, 245]). In this classical approach a definable concept (set) is a relation on the domain of a given structure whose elements are precisely those elements satisfying some formula in the structure. Semantics of such formula enables to determine precisely for a given element (object) whether it belongs to the concept or not. However, the issue of definability of concepts is somewhat complicated by the pervasive presence of vagueness and ambiguity in natural language (see [126, 127, 244]). Therefore, in numerous applications, the concepts of interest may only be defined approximately on the basis of available, incomplete, imprecise or noisy information about them, represented, *e.g.*, by positive and negative examples (see [6, 7, 8, 9, 10, 11, 12, 13]). Such concepts are often called *vague* (imprecise) concepts. We say that a concept is vague when there may be cases (elements, objects) in which there is no clear fact of the matter whether the concept applies or not. Hence, the classical approach to concept definability known from classical logic cannot be applied for vague concepts. At the same time an approximation of a vague concept consists in construction of an algorithm (called a classifier) for this concept, which may be treated as a constructive, approximate description of the concept. This description enables to classify testing objects, that is, to determine for a given object whether it belongs to the concept approximated or not to which degree.

There is a long debate in philosophy on vague concepts (see, *e.g.*, [126, 127, 128]) and recently computer scientists (see, *e.g.*, [79, 82, 83, 246, 247, 248, 249]) as well as other researchers have become interested in vague concepts. Since the classical approach to concept definability known from classical logic cannot be applied for vague concepts new methods of definability have been proposed. Professor Lotfi Zadeh (see [250]) introduced a very successful approach to definability of vague concepts. In this approach, sets are defined by partial membership in contrast to crisp membership used in the classical definition of a set. Rough set theory proposed a method of concept definability by employing the lower and upper approximation, and the boundary region of this concept (see Section 2.1). If the boundary region of a set is empty it means that a particular set is crisp, otherwise the set is rough (inexact). The non-empty boundary region of the set means that our knowledge about the set is not sufficient to define the set precisely. Using the lower and upper approximation, and the boundary region of a given concept a classifier can be constructed. Assume there is given a decision table $\mathbf{A} = (U, A, d)$, whose binary decision attribute with values 1 and 0 partitions the set of objects in two disjoint ones: C and C'. The set C contains objects with the decision attribute value equal to 1, and the set C' contains objects with the decision attribute value equal to 0. The sets C and C' may also be interpreted in such a way that the set C is a certain concept to be approximated and the set C' is the complement of this concept ($C' = U \setminus C$). If we define for concept C and its complement C', their A-lower approximations $\underline{A}C$ and $\underline{A}C'$, the A-upper approximation $\overline{A}C$, and the A-boundary $BN_A(C)$ ($BN_A(C) = \overline{A}C \setminus \underline{A}C$), we obtain a simple classifier which operates in such a way that a given testing object u is classified to concept C if it belongs to the lower approximation $\underline{A}C$. Otherwise, if object u belongs to the lower approximation $\underline{A}C'$, it is classified to the complement of concept C. However, if the object belongs neither to $\underline{A}C$ nor $\underline{A}C'$, but it belongs to $BN_A(C)$, then the classifier cannot make an unambiguous decision about membership of the object, and it has to respond that the object under testing simultaneously belongs to the concept C and its complement C', which means it is a border object. In this case the membership degree of a tested object $u \in U$ to concept $C \subseteq U$ is expressed numerically with the help of a *rough membership function* (see, *e.g.*, [16, 17]). The *rough membership function* μ_C quantifies the degree of relative overlap between the concept C and the equivalence class to which u belongs. It is defined as follows:

$$\mu_C(u) : U \to [0,1] \text{ and } \mu_C(u) = \frac{card([u]_{IND_{\mathbf{A}}(A)} \cap C)}{card([u]_{IND_{\mathbf{A}}(A)})}.$$

As we can see, in order to work the classifier described above, it is necessary for the tested object to belong to one of the equivalence classes of relation $IND_{\mathbf{A}}(A)$. However, there is one more instance remaining when the tested object does not belong to any equivalence class of relation $IND_{\mathbf{A}}(A)$. In such case, the classifier under consideration cannot make any decision about membership of the tested object and has to say: *"I do not know"*.

Unfortunately, the case when the tested object does not belong to any equivalence class of relation $IND_{\mathbf{A}}(A)$ frequently occurs in practical applications. It is due to the fact that if the objects under testing do not belong to the decision table that was known at the beginning, but to its extension, the chances are small that in a given decision table, there exists an object (called a training object) whose conditional attribute values are identical to those in the testing object. However, it follows from the definitions of the relation $IND_{\mathbf{A}}(A)$ that the testing object for which there is no training object cannot be classified by the classifier described above. In such a case, one can say that the *extension* of this classifier is very small. For the above reason, the classic approach to classifying objects in the rough set theory (described above) requires generalization.

It is worth noticing that in machine learning and pattern recognition (see, *e.g.*, [6, 8, 9, 10, 11, 12, 13]), this issue is known under the term *learning concepts by examples* (see, *e.g.*, [10]). The main problem of learning concepts by examples is that the description of a concept under examination needs to be created on the basis of known examples of that concept. By creating a concept description we understand detecting such properties of exemplary objects belonging to this concept that enable further examination of examples in terms of their membership in the concept under examination. A natural way to solve the problem of learning concepts by examples is *inductive reasoning* (see, *e.g.*, [251, 252]). In inductive reasoning we assume as true the sentence stating a general regularity, at the same time we do that on the basis of acknowledging sentences stating individual instances of this regularity (see, *e.g.*, [251, 252]). This is the reasoning according to which decisions in the real world are often made relying on incomplete or even flawed information. This takes place in the cases of answers to questions connected with forecasting, checking hypotheses or making decisions.

In the case of the problem of learning concepts by examples, the usage of inductive reasoning means that while obtaining further examples of objects belonging to the concept (the so-called positive examples) and examples of objects not belonging to the concept (the so-called negative examples), an attempt is made to find such description that correctly matches all or almost all examples of the concept under examination.

From the theoretical point of view, in the rough set theory the classic approach to concept approximation was generalized by Professor Skowron and Professor Stepaniuk (see [253]). This approach is consistent with the philosophical view (see, *e.g.*, [126, 127]) and the logical view (see, *e.g.*, [128]). The main element of this generalization is an *approximation space*. The approximation space (see, *e.g.*, [246, 253, 254, 255]) is a tuple $AS = (U, I, \nu)$, where

– U is a non-empty set of objects,
– $I : U \to P(U)$ is an *uncertainty function* and $P(U)$ denotes the powerset of U,
– $\nu : P(U) \times P(U) \to [0, 1]$ is a *rough inclusion function*.

The uncertainty function I defines for every object $u \in U$ a set of objects indistinguishable with u or similar to u. The set $I(u)$ is called the neighborhood of u. If U is a set of objects of a certain decision table $\mathbf{A} = (U, A, d)$, then in the simplest case the set $I(u)$ may be the equivalence class $[u]_{IND_{\mathbf{A}}(A)}$. However, in a general case the set $I(u)$ is usually defined with the help of a special language such as GDL or NL (see Section 4.7).

The rough inclusion function ν defines the degree of inclusion of X in Y, where $X, Y \subseteq U$. In the simplest case, rough inclusion can be defined by:

$$\nu(X,Y) = \begin{cases} \frac{card(X \cap Y)}{card(X)} & \text{if } X \neq \emptyset \\ 1 & \text{if } X = \emptyset. \end{cases}$$

This measure is widely used by the data mining and rough set communities (see, *e.g.*, [16, 17, 246, 253]). However, rough inclusion can have a much more general form than inclusion of sets to a degree (see [192, 247, 249]).

It is worth noticing that in literature (see, *e.g.*, [247]) a *parameterized approximation space* is considered instead of the approximation space. Any parameterized approximation space consists of a family of approximation spaces creating the search space for data models. Any approximation space in this family is distinguished by some parameters. Searching strategies for optimal (sub-optimal) parameters are basic rough set tools in searching for data models and knowledge. There are two main types of parameters. The first ones are used to define object sets (neighborhoods), the second are measuring the inclusion or closeness of neighborhoods.

For an approximation space $AS = (U, I, \nu)$ and any subset $X \subseteq U$ the lower and the upper approximations are defined by:

- $LOW\left(AS, X\right) = \left\{u \in U : \nu\left(I\left(u\right), X\right) = 1\right\}$,
- $UPP\left(AS, X\right) = \left\{u \in U : \nu\left(I\left(u\right), X\right) > 0\right\}$, respectively.

The lower approximation of a set X with respect to the approximation space AS is the set of all objects which can be classified with certainty as object of X with respect to AS. The upper approximation of a set X with respect to the approximation space AS is the set of all objects which can be possibly classified as objects of X with respect to AS.

Several known approaches to concept approximations can be covered using the approximation spaces discussed here, *e.g.*, the approach given in [16, 17], approximations based on the variable precision rough set model (see, *e.g.*, [256]) or tolerance (similarity) rough set approximations (see, *e.g.*, [253]).

Similarly to the classic approach, the lower and upper approximation in the approximation space AS for a given concept C may be used to classify objects to this concept. In order to do this one may examine the membership of the tested objects to $LOW\left(AS, C\right)$, $LOW\left(AS, C'\right)$ and $UPP\left(AS, C\right) \setminus LOW\left(AS, C\right)$.

However, in machine learning and pattern recognition (see, *e.g.*, [6, 8, 9, 10, 11, 12, 13]), we often search for approximation of a concept $C \subseteq U^*$ in an approximation space $AS^* = (U^*, I^*, \nu^*)$ having only a partial information about AS^* and C, i.e., information restricted to a sample $U \subseteq U^*$. Let us denote the restriction of AS^* to U by $AS = (U, I, \nu)$, i.e., $I(x) = I^*(x) \cap U$, $\nu(X,Y) = \nu^*(X,Y)$ for $x \in U$, and $X, Y \subseteq U$ (see Fig. 3).

To decide if a given object $u \in U^*$ belongs to the lower approximation or to the upper approximation of $C \subseteq U^*$, it is necessary to know the value $\nu^*(I^*(u), C)$. However, in the case there is only partial information about the approximation space AS^* available, one must make an estimation of such a value $\nu^*(I^*(u), C)$

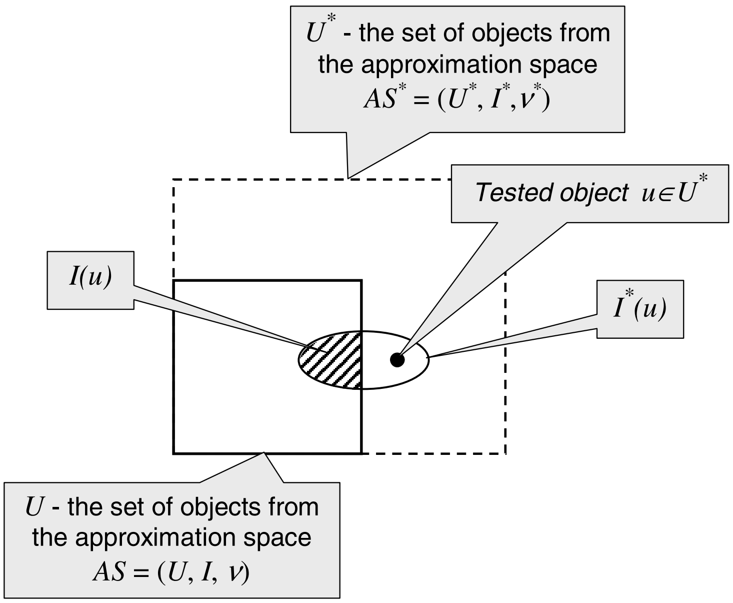

U^* - the set of objects from the approximation space $AS^* = (U^*, I^*, \nu^*)$
Tested object $u \in U^*$
$I(u)$
$I^*(u)$
U - the set of objects from the approximation space $AS = (U, I, \nu)$

Fig. 3. An approximation space AS and its extension AS^*

rather than its exact value. In machine learning, pattern recognition or data mining, different heuristics are used for estimation of the values of ν^*. Using different heuristic strategies, values of another function ν' are computed and they are used for estimation of values of ν^*. Then, the function ν' is used for deciding if objects belong to C or not. Hence, we define an approximation of C in the approximation space $AS' = (U^*, I^*, \nu')$ rather than in $AS^* = (U^*, I^*, \nu^*)$. Usually, it is required that the approximations of $C \cap U$ in AS and AS' are close (or the same).

The approach presented above (see, *e.g.*, [83, 246, 248, 249]) became an inspiration for finding out of a number of methods which would enable to enlarge the extension of constructed classifiers, that is, to make the classifiers under construction to be able to classify any objects, and not only those belonging to a given decision table.

Some other issues concerning the rough set approach to vague concept approximation are discussed, e.g., in [83, 128, 248, 249]. Among these issues are the higher order vagueness (i.e., nondefinability of boundary regions), adaptive learning of concept approximation, concept drift, and sorites paradoxes.

One of the basic ways of increasing the extension of classifiers is to approximate the concepts not with the help of the equivalence class of relation IND (see above) but with the help of the patterns of the established language which different objects may match, both from the training table and its extension. A given object matches the pattern if it is compatible with the description of this pattern. Usually, the pattern is constructed in such a way that all or almost all its matching objects belong to the concept under study (the decision class). Moreover, it is required that the objects from many equivalence classes of relation IND could match the patterns. Thus, the extension of classifiers based on patterns is dramatically greater than the extension of classifiers working on the basis of equivalence classes of relation IND. These types of patterns are often called *decision rules* (see Section 2.3). In literature one may encounter many methods of computing decision rules from data and methods enabling preprocessing the data in order to construct effective classifiers. Into this type of methods one may include, for example, discretization of attribute values (see Section 2.2), methods computing decision rules (see Section 2.3), shortening and generalization of decision rules (see Section 2.5).

The determined decision rules may be applied to classifiers construction. For instance, let us examine the situation, when a classifier is created on the basis of decision rules from the set $RUL(\mathbf{A})$ computed for a given decision table $\mathbf{A} = (U, A, d)$, and at the same time decision attribute d describes the membership to a certain concept C and its complement C'. [1]

The set of rules $RUL(\mathbf{A})$ is the sum of two subsets $RUL(\mathbf{A}, C)$ and $RUL(\mathbf{A}, C')$, where $RUL(\mathbf{A}, C)$ is the set of rules classifying objects to C and $RUL(\mathbf{A}, C')$ is a set of rules classifying objects to C'. For any tested object u, by $MRul(\mathbf{A}, C, u) \subseteq RUL(\mathbf{A}, C)$ and $MRul(\mathbf{A}, C', u) \subseteq RUL(\mathbf{A}, C')$ we denote sets of such rules whose predecessors match object u and classify objects to C and C', respectively.

Let $AS = (U, I, \nu)$ be an approximation space, where:

1. $\forall u \in U : I(u) = \bigcup_{r \in MRul(\mathbf{A},C,u)} Supp_{\mathbf{A}}(r) \cup \bigcup_{r \in MRul(\mathbf{A},C',u)} Supp_{\mathbf{A}}(r)$
2. $\forall X, Y \subseteq U : \nu(X, Y) = \begin{cases} \frac{card(X \cap Y)}{card(X)} & \text{if } X \neq \emptyset \\ 1 & \text{if } X = \emptyset. \end{cases}$

The above approximation space AS may be extended in a natural way to approximation space $AS' = (U^*, I^*, \nu')$, where:

1. $I^* : U^* \longrightarrow P(U^*)$ such that $\forall u \in U : I^*(u) = I(u)$,
2. $\forall X, Y \subseteq U^* : \nu'(X, Y) = \begin{cases} \frac{card(X \cap Y)}{card(X)} & \text{if } X \neq \emptyset \\ 1 & \text{if } X = \emptyset. \end{cases}$

Let us notice that such a simple generalization of functions I to I^* and ν to ν' is possible because function I may determine the neighborhood for a given object belonging to U^*. It results from the fact that decision rules from set $RUL(\mathbf{A})$ may recognize objects not only from set U but also from set $U^* \setminus U$. Approximation space AS' may now also be used to construct a classifier which classifies objects from set U^* to concept C or its complement C'. In creating such a classifier the key problem is to resolve the conflict between the rules

[1] For simplicity of reasoning we consider only binary classifiers, i.e. classifiers with two decision classes. One can easily extend the approach to the case of classifiers with more decision classes.

classifying the tested object to the concept or to its complement. Let us notice that this conflict occurs because in practice we do not know function ν^* but only its approximation ν'. That is why, there may exist such a tested object u_t that the values $\nu'(\{u_t\}, C)$ and $\nu'(\{u_t\}, C')$ are high (that is close to 1), while values $\nu^*(\{u_t\}, C)$ and $\nu^*(\{u_t\}, C')$ are very different (*e.g.*, $\nu^*(\{u_t\}, C)$ is close to 1 and $\nu^*(\{u_t\}, C')$ is close to 0).

Below, we present the definition of such a classifier in the form of a function that returns the value YES when the tested object belongs to C or the value NO when the tested object belongs to C':

$$\forall u \in U : Classifier(u) = \begin{cases} YES & \text{if } \nu'(\{u\}, C) > 0.5 \\ NO & \text{otherwise.} \end{cases} \tag{5}$$

Obviously, other rough inclusion functions may be defined (see, *e.g.*, [192, 247, 249]). Thus, we obtain different classifiers. Unfortunately, a classifier defined with the help of Equation (5) is impractical because the function ν' used in it does not introduce additional parameters which enable to recognize of objects to the concept and its complement whereas in practical applications in constructing classifiers based on decision rules, functions are applied which give the strength (weight) of the classification of a given tested object to concept C or its complement C' (see, *e.g.*, [196, 199, 216, 217, 241]). Below, we present a few instances of such weights (see [199]).

1. A *simple strength* of decision rule set is defined by

$$SimpleStrength(C, u_t) = \frac{card(MRul(\mathbf{A}, C, u_t))}{card(RUL(\mathbf{A}, C))}.$$

2. A *maximal strength* of decision rule set is defined by

$$MaximalStrength(C, u_t) = max_{r \in MRul(\mathbf{A}, C, u_t)} \left\{ \frac{Supp_{\mathbf{A}}(r)}{card(C)} \right\}.$$

3. A *basic strength* or *a standard strength* of decision rule set is defined by

$$BasicStrength(C, u_t) = \frac{\sum\limits_{r \in MRul(\mathbf{A}, C, u_t)} Supp_{\mathbf{A}}(r)}{\sum\limits_{r \in RUL(\mathbf{A}, C)} Supp_{\mathbf{A}}(r)}.$$

4. A *global strength* of decision rule set is defined by

$$GlobalStrength(C, u_t) = \frac{card\left(\bigcup\limits_{r \in MRul(\mathbf{A}, C, u_t)} Supp_{\mathbf{A}}(r) \right)}{card(C)}.$$

Using each of the above rules weight, a rough inclusion function corresponding to it may be defined. Let us mark any established weight of rule sets as S. For weight S we define an exemplary rough inclusion function ν_S in the following way:

$$\forall X, Y \subseteq U : \nu_S(X,Y) = \begin{cases} 0 & \text{if } Y = \emptyset \wedge X \neq \emptyset \\ 1 & \text{if } X = \emptyset \\ \frac{S(Y,u)}{S(Y,u)+S(U\setminus Y,u)} & \text{if } X = \{u\} \text{ and } \\ & \quad S(Y,u) + S(U \setminus Y, u) \neq 0 \\ \frac{1}{2} & \text{if } X = \{u\} \text{ and } \\ & \quad S(Y,u) + S(U \setminus Y, u) = 0 \\ \frac{\sum_{u \in X} \nu_S(\{u\},Y)}{card(X)} & \text{if } card(X) > 1 \end{cases},$$

where for an established set Y and object u the weights $S(Y,u)$ and $S(U \setminus Y, u)$ are computed using the decision rule set generated for table $\mathbf{A} = (U, A, d_Y)$, where attribute d_Y describes the membership of objects from U to the set Y.

The rough inclusion function defined above may be used to construct the classifier as it is done in Equation (5). Such a classifier executes a simple negotiation method between the rules classifying the tested object to the concept and rules classifying the tested object to the complement of the concept (see Section 2.6). It simply is based on classifying tested object u to concept C only when with the established weight of rule sets S the value $\nu_S(\{u\}, C)$ is bigger than $\nu_S(\{u\}, C')$. Otherwise, object u is classified to the complement of concept C.

In this paper, the weight *BasicStrength* is used in experiments related to construction of classifiers based on decision rules to resolve conflicts between rule sets.

2.9 Evaluation of Classifiers

In order to evaluate the classifier quality in relation to the data analyzed, a given decision table is partitioned into the two tables in a general case (see, *e.g.*, [11, 257, 258]):

1. *the training table* containing objects on the basis of which the algorithm learns to classify objects to decision classes,
2. *the test table*, by means of which the classifier learned on the training table may be evaluated when classifying all objects belonging to this table.

The numerical measure of the classifier evaluation is often the number of mistakes made by the classifier during classification of objects from the test table in comparison to all objects under classification (*the error rate*, see, *e.g.*, [11, 196, 198]). However, the method of the numerical classifier evaluation, used most often, is the method based on a *confusion matrix*. The *confusion matrix* (see, *e.g.*, [15, 257, 259]) contains information about actual and predicted classifications done by a classifier. Performance of such systems is commonly evaluated using the data in the matrix. The Table 1 shows the confusion matrix for a two class classifier, i.e., for a classifier constructed for a concept.

Table 1. The confusion matrix

		Predicted	
		Negative	Positive
Actual	Negative	*TN*	*FP*
	Positive	*FN*	*TP*

The entries in the confusion matrix have the following meaning in the context of our study (see, *e.g.*, [260]):

- TN (*True Negatives*) is the number of correct predictions that an object is a negative example of a concept of the test table,
- FP (*False Positives*) is the number of incorrect predictions that an object is a positive example of a concept of the test table,
- FN (*False Negatives*) is the number of incorrect predictions that an object is a negative example of a concept of the test table,
- TP (*True Positives*) is the number of correct predictions that an object is a positive example of a concept of the test table.

Several standard terms (parameters) have been defined for the two class confusion matrix:

- the *accuracy* (ACC) defined for a given classifier by the following equality:

$$ACC = \frac{TN + TP}{TN + FN + FP + TP},$$

- the *accuracy for positive examples* or the *sensitivity* (see, *e.g.*, [260]) or the *true positive rate* (TPR) (see, *e.g.*, [257]) defined for a given classifier by the following equality:

$$TPR = \frac{TP}{TP + FN},$$

- the *accuracy for negative examples* or the *specificity* (see, *e.g.*, [260]) or the *true negative rate* (TNR) (see, *e.g.*, [257]) defined for a given classifier by the following equality:

$$TNR = \frac{TN}{TN + FP}.$$

An essential parameter is also the number of classified objects from the test table in comparison to the number of all objects from this table since classifiers may not always be able to classify the objects. This parameter, called the *coverage* (see, *e.g.*, [11, 15]), may be treated as an extension measure of the classifier. Thus, in order to evaluate classifiers, also the following numerical parameters are applied in this paper:

1. the *coverage* (COV) defined for a given classifier by the following equality:

$$COV = \frac{TN + FP + FN + TP}{\text{the number of all objects of the test table}},$$

2. the *coverage for positive examples* ($PCOV$) defined for a given classifier by the following equality:

$$PCOV = \frac{FN + TP}{\text{the number of all positive examples of a concept of the table}},$$

3. the *coverage for negative examples* ($NCOV$) defined for a given classifier by the following equality:

$$NCOV = \frac{TN + FP}{\text{the number of all negative examples of a concept of the table}},$$

4. the *real accuracy* defined for a given classifier by: $ACC \cdot COV$,
5. the *real accuracy for positive examples* or the *real true positive rate* defined for a given classifier by: $TPR \cdot PCOV$,
6. the *real accuracy for negative examples* or the *real true negative rate* defined for a given classifier by: $TNR \cdot NCOV$.

Besides that, in order to evaluate classifiers still different parameters are applied. These are, for instance, time of construction of a classifier on the basis of a training table or the complexity degree of the classifier under construction (*e.g.*, the number of generated decision rules).

In summary, in this paper the main parameters applied to the evaluation of classifiers are: the accuracy, the coverage, the real accuracy, the accuracy for positive examples, the coverage for positive examples, the real accuracy for positive examples, the accuracy for negative examples, the coverage for negative examples and the real accuracy for negative examples.

They are used in experiments with AR schemes (see Section 5.8) and experiments related to detecting behavioral patterns (see Section 6.25 and Section 6.26). However, in experiments with automated planning another method of classifier quality evaluation was applied (see Section 7.21). It results from the fact that this case is about automated generating the value of complex decision that is a plan which is a sequence of actions alternated with states. Hence, to compare this type of complex decision values the above mentioned parameters may not be used. Therefore, to compare the plans generated automatically with the plans available in the data set we use a special classifier based on concept ontology which shows the similarity between any pair of plans (see Section 7.18).

It is worth noticing that in literature there may be found another, frequently applied method of measuring the quality of created classifiers. It is a method based on ROC curve (*Receiver Operating Characteristic curve*) (see, *e.g.*, [260, 261, 262]). This method is available, for instance, in system ROSETTA (see, *e.g.*, [259, 263, 264]). It is also worthwhile mentioning that the author of this paper participated in construction of programming library RSES-lib, creating the computational kernel of system ROSETTA (see [230, 259] for more details).

In order not to make the value of the determined parameter of the classifier evaluation depending on a specific partitioning the whole decision table into a training and test parts, a number of methods are applied which perform tests to determine which parameter values of the classifier evaluation are creditable.

The methods of this type applied most often are *train-and-test* and *cross-validation* (see, *e.g.*, [11, 258, 265]). The *train-and-test* method is usually applied to decision tables having at least 1000 objects (see, *e.g.*, [11]). It consists in a random isolation of two test subtables from the whole data available, treating one of them as a training subtable and the other as a test subtable. The training and test subtables are usually separated (although not always) and altogether make the available decision table. It is crucial, however, that at least some part of the objects from the test subtable does not occur in the training subtable. The proportion between the number of objects in the test and training subtables depends on a given experiment but it is usually such that the number of objects in the test part constitutes from 20 to 50 percent of the number of objects in the whole data available (see, *e.g.*, [11]). The *cross-validation* method is applied to evaluate a classifier when the number of objects in the decision table is less than 1000 (see, *e.g.*, [11]). This method consists in partitioning data in a random way into m equal parts and, then performing m experiments with them. In each of these experiments, a local coefficient of the classifier evaluation is calculated for a situation when one of the parts into which the data was divided is a set of tested objects, and the remaining $m - 1$ parts (temporarily combined) are treated as a set of training objects. Finally, a coefficient of the classifier evaluation as an average arithmetical coefficient of all experiments is calculated. The number m is determined depending on the specific data and should be selected in such a way that the test parts not to have too few objects. In practice, m is an integer ranging from 5 to 15 (see, *e.g.*, [11]).

All decision tables used in experiments have more than 1000 objects, in this paper. That is why in order to determine the parameter of the classifier quality the *train-and-test* method is always applied. Moreover, each experiment is repeated 10 times for ten random partitions into two separate tables (training and test). Hence, the result of each experiment is the arithmetical mean obtained from the results of its repetitions. Additionally, the standard deviation of the received result is given.

2.10 Problem of Low Coverage

If a given tested object matches the predecessor of a certain basic decision rule (that is the values of condition attributes of this object are the same as the values of the descriptors from the rule predecessor corresponding to them), then this rule may be used to classify this object, that is, the object is classified to the decision class occurring in the rule successor. In this case we also say that a given tested object is recognized by a certain decision rule. However, if a given tested object is recognized by different decision rules which classify it to more than one decision classes, then negotiation methods between rules are applied (see Section 2.6 and Section 2.8).

In practice, it may happen that a given tested object does not match the predecessor of any of the available decision rules. We say that this object is not recognized by a given classifier based on decision rules and what follows it cannot be classified by this classifier. It is an unfavorable situation, for we often expect from the classifiers to classify all or almost all tested objects. If there are many of the unclassified objects, then we say that a given classifier has too low an extension. It is expressed numerically by a low value of the coverage parameter (see Section 2.9).

A number of approaches which enable to avoid a low coverage of classifiers based on decision rules were described in literature. They are for example:

1. The application of classifiers based on the set of all rules with a minimum number of descriptors (see Section 2.4) which usually have a high extension (see, *e.g.*, [196, 198]).
2. The application of rule classifiers constructed on the basis of covering algorithms and partial matching mechanism of the objects to the rules (see, *e.g.*, [10, 213, 214, 216, 217, 222, 223, 266]).
3. The application of classifiers based on decision rules which underwent the process of generalization of rules owing to which the classifier extension usually increases (see Section 2.5).
4. The application of classifiers based on a lazy learning which does not require preliminary computation of decision rules, for decision rules needed for object classification are discovered directly in a given decision table during the classification of the tested object (see, *e.g.*, [197, 198, 267]).

All the methods mentioned above have their advantages and disadvantages. The first method has an exponential time complexity which results from the complexity of the algorithm computing all reducts (see Section 2.4). The second method is very quick, for it is based on rules computed with the help of the covering method. However, in this method there are often applied approximation rules to classify objects (determined as a result of a partial matching objects to the rules). Therefore, the quality of classification on the basis of such rules may be unsatisfactory. The third method uses the operation of rule generalization. Owing to this operation the extension of the obtained rules increases. However, it does not lead to such a high extension as in the case of the first, second and fourth method. Apart from that the operation of rule generalization is quite time consuming. Whereas, the fourth method, although does not require preliminary computation of decision rules, its pessimistic computational time complexity of each tested object classification is of order $O(n^2 \cdot m)$, where n is the number of objects in the training table and m is the number of condition attributes. Hence, for bigger decision tables this method cannot be applied effectively.

There is one more possibility remaining to build classifiers on the basis of rules computed with the covering method without using partial matching tested objects to the rules. Obviously, classifiers based on such rules may have a low coverage. However, they usually have a high quality of the classification. It is extremely crucial in many applications (for example in medical and financial ones) where it is required that the decisions generated by classifiers be always or almost always correct. In such applications it is sometimes better for the classifier to say *I do not know* rather than make a wrong decision. That is why in this paper we use classifiers based on rules computed with covering method (without partial matching objects to the rules) agreeing on a low coverage of such classifiers in cases when classifiers based on the set of all rules with minimum number of descriptors cannot be applied (too large analyzed decision tables).

3 Methods of Constructing Stratifying Classifiers

The algorithm of concept approximation, presented in Subsection 2.8, consists in classifying the tested objects to the lower approximation of this concept, the lower approximation of complement of this concept or its border. Many methods enabling increase of the extension of classifiers under construction, in rough set theory are proposed (see Section 2.8). Discretization of attribute values (see Section 2.2), methods of calculating and modifying decision rules (see Sections 2.3, 2.4, 2.5), and partial matching method (see Section 2.10) are examples of such methods. As a result of applying these methods, there are constructed classifiers able to classify almost every tested object to the concept or its complement.

At first glance this state of affairs should dispose optimistically for approximation methods can be expanded for tested objects from beyond a given decision table, which is necessary in inductive learning (see Section 2.8). Unfortunately, such a process of generalizing concept approximation encounters difficulties in classifying new (unknown during the classifier learning) tested objects. Namely, after expanding the set of objects U of a given information system with new objects, equivalence classes of these objects are often disjoint with U. This means that if such objects match the description of a given concept C constructed on the basis of set U, this match is often incidental. Indeed due to the unfamiliarity the process generalization of decision rules may go too far (*e.g.*, decision rules are too short) because of absence of these new objects when the concept description was created. It may happen that the properties (attributes) used to describe a concept are chosen in wrong way. So, if a certain tested object from outside the decision table is classified, then it may turn out that, in the light of the knowledge gathered in a given decision table, this object should be classified neither to the concept nor to its complement but to the concept border, which expresses our uncertainty about the classification of this object. Meanwhile, most of the classifiers currently constructed classify the object to the concept or its complement. A need of use the knowledge from a given table arises in order to determine the coefficient of certainty that the object under testing belongs to the approximated concept. In other words, we would like to determine, with reference to the tested object, how certain the fact is that this object belongs to the concept. And at the same time it would be the best to express if the certainty coefficient by a number, *e.g.*, from $[0, 1]$. In literature such a numerical coefficient is expressed using different kinds of rough membership functions (see Section 2.8). If a method of determining such a coefficient is given, it may be assumed that the coefficient values are discretized which leads to a sequence of concept layers arranged linearly. The first layer in this sequence represents objects which, without any doubt do not belong to the concept (the lower approximation of the concept complement). The next layers in the sequence represent objects belonging to the

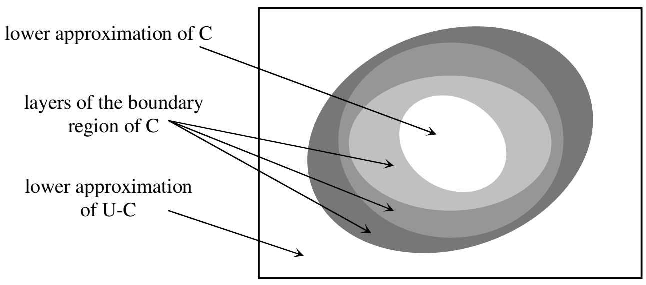

lower approximation of C
layers of the boundary region of C
lower approximation of U-C

Fig. 4. Layers of a given concept C

concept more and more certainly (border layers of the concept). The last layer in this sequence represents objects certainly belonging to the concept, that is, the ones belonging to the lower concept approximation (see Fig. 4).

Let us add that this type of concept layers may be defined both on the basis of the knowledge gathered in data tables and using additional domain knowledge provided by experts.

3.1 Stratifying Classifier

In order to examine the membership of tested objects to individual concept layers, such classifiers are needed that can approximate all layers of a given concept at the same time. Such classifiers are called in this paper *stratifying classifiers.*

Definition 1 (*A stratifying classifier*). *Let* $\mathbf{A} = (U, A, d)$ *be a decision table whose objects are positive and negative examples of a concept C (described by a binary attribute d).*

1. *A partition of the set* U *is a family* $\{U_1, ..., U_k\}$ *of non-empty subsets of the set* U *(where* $k > 1$*) such that the following two conditions are satisfied:*
   (a) $U = U_1 \cup ... \cup U_k$,
   (b) $\forall_{i \neq j}\ U_i \cap U_j = \emptyset$.
2. *A partition of the set* U *into a family* $U_C^1, ..., U_C^k$ *we call the partition of* U *into layers in relation to concept* C *when the following three conditions are satisfied:*
   (a) set U_C^1 *includes objects which, according to an expert, certainly do not belong to concept* C *(so they belong to a lower approximation of its complement),*
   (b) for every two sets U_C^i, U_C^j *(where* $i < j$*), set* U_C^i *includes objects which, according to an expert, belong to concept* C *with a degree of certainty lower than the degree of certainly of membership of objects of* U_C^j *in* U,
   (c) set U_C^k *includes objects which, according to an expert, certainly belong to concept* C*, viz., to its lower approximation.*
3. *Each algorithm which assigns (classifies) tested objects into one of the layers belonging to a partition of the set* U *in relation to the concept* C*, is called a stratifying classifier of the concept* C.

4. *In practice, instead of using layer markings $U_C^1, ..., U_C^k$, elements of the set $E = \{e_1, ..., e_k\}$ are used to label layers, whereas the stratifying classifier constructed for the concept C which classifies each tested object into one of the layers labeled with labels from the set E, is denoted by μ_C^E.*
5. *If the stratifying classifier μ_C^E classifies a tested object u into the layer labeled by $e \in E$, then this fact is denoted by the equality $\mu_C^E(u) = e$.*

An expert may divide the set of objects U into layers in two following ways:

1. by an assignment of weight labels to all training objects arbitrary (see Section 3.2),
2. by providing heuristics which may be applied in construction of a stratifying classifier (see Section 3.3).

Stratifying classifiers can be very useful when we need to estimate realistically what the certainty of membership of a tested object to a concept is, without determining whether the object belongs to the concept or not. Apart from that, stratifying classifiers may be used to construct the so-called production rules (see Section 5.3).

In the paper, two general ways of construction of stratifying classifiers are presented. The first one is the *expert approach* consisting in defining by an expert an additional attribute in data which describes the membership of objects to particular layers of the concept. Next, a classifier differentiating layers as decision classes is built (see Section 3.2).

The second approach is called the *automatic approach* and is based on designing algorithms which are extensions of classifiers enabling the classification of objects into layers of a concept on the basis of certain premises and experimental observations (see Section 3.3).

3.2 Stratifying Classifiers Based on the Expert Approach

In construction of stratifying classifiers using expert knowledge, it is assumed that for all training objects not only the binary classification of training objects to a concept or outside the concept is known but we also know the assignment of all training objects into the specific concept layers. Using this approach an additional knowledge needs to be gained from a domain knowledge. Owing to that a classical classifier may be built (*e.g.*, the one based on a set of rules with a minimal number of descriptors) which directly classifies the objects to different concept layers. This classifier is built on the basis of a decision attribute which has as many values as many concept layers are there, and each of these values is a label of one of the layers.

3.3 Stratifying Classifiers Based on the Automatic Approach

In construction of stratifying classifiers using the automatic approach, the assignment of all training objects to specific concept layers is unknown but we only know the binary classification of training objects to a concept or its complement. However, the performance of a stratifying classifier is, in this case, connected with a certain heuristics which supports discernibility of objects belonging to a lesser or greater degree to the concept, that is, objects belonging to different layers of this concept. Such a heuristic determines the way an object is classified to different layers and, thus, it is called a *stratifying heuristic*.

Many different types of heuristics stratifying concepts may be proposed. These may be, *e.g.*, heuristics based on the difference of weights of decision rules classifying tested objects to concept and its complement or heuristics using a k-NN algorithm of k nearest neighbors (compare with [78, 200, 268]). In this paper, however, we are concerned with a new type of stratifying heuristics using the operation of decision rule shortening (see Section 2.5).

The starting point of the presented heuristics is the following observation. Let us assume that for a certain consistent decision table $\mathbf{A}$ whose decision is a binary attribute with values 1 (objects belonging to the concept C) and 2 (objects belonging to the complement of concept C which is denoted by C'), a set of decision rules, $RUL(\mathbf{A})$ was calculated. The set $RUL(\mathbf{A})$ is the sum of two separate subsets of rules $RUL_1(\mathbf{A})$ (classifying objects to C) and $RUL_2(\mathbf{A})$ (classifying objects to C'). Now, let us shorten the decision rules from $RUL_1(\mathbf{A})$ to obtain the coefficient of consistency equal to 0.9 by placing the shortened decision rules in the set $RUL_1(\mathbf{A}, 0.9)$. Next, let $RUL'(\mathbf{A}) = RUL_1(\mathbf{A}, 0.9) \cup RUL_2(\mathbf{A})$. In this way, we have increased the extension of the input decision set of rules $RUL(\mathbf{A})$ in relation to the concept C, viz., as a result of shortening of the rules, the chance is increased that a given tested object is recognized by the rules classifying to the concept C. In other words, the classifier based on the set of rules $RUL'(\mathbf{A})$ classifies objects to the concept C more often.

Now, if a certain tested object u, not belonging to table $\mathbf{A}$, is classified to C' by the classifier based on the rule set $RUL'(\mathbf{A})$, then the chance that object u actually belongs to C' is much bigger than in the case of using the set of rules $RUL(\mathbf{A})$. The reason for this assumption is the fact that it is harder for a classifier based on the set of rules $RUL'(\mathbf{A})$ to classify objects to C' for the rules classifying objects to C are shortened in it and owing to that they recognize the objects more often. If, however, an object u is classified to C', then some of its crucial properties identified by the rules classifying it to C' must determine this decision. If shortening of the decision rules is greater (to the lower consistency coefficient), then the change in the rule set extension will be even bigger.

Summing up the above discussion, we conclude that rule shortening makes it possible to change the extensions of decision rule sets in relation to chosen concepts (decision classes), and owing to that one can obtain a certain type of approximation based on the certainty degree, concerning the membership of tested objects to the concept under consideration where different layers of the concept are modeled by applying different coefficients of rule shortening.

In construction of algorithms producing stratifying classifiers based on shortening of decision rules, there occurs a problem with the selection of accuracy coefficient threshold to which decision rules are shortened. In other words, what we mean here is the range and the step with which the accuracy coefficient threshold must be selected in order to obtain sets of rules enabling an effective description of the actual layers of the concept approximated. On the basis of previous experimental experience (see, *e.g.*, [196, 198]), in this paper, we establish that the shortening thresholds of decision rule consistency coefficient are selected from the range 0.5 to 1.0. The lower threshold limit (that is, 0.5) results from the experimental observation that if we shorten rules classifying objects to a certain concept C below the limit 0.5 (without simultaneous shortening of rules classifying objects to C'), then although their extension increases dramatically (they classify objects to the concept C very often), their certainty falls to an absolutely unsatisfactory level. However, the upper limit of the threshold (that is 1.0) simply means leaving only accurate rules in the set of rules, and rejecting other approximation rules which could have occurred for a given decision table.

If it comes, however, to the change step of the chosen threshold of the rule coefficient of consistency, we set it at 0.1. This change step is dictated by the fact that it enables a general search of thresholds from 0.5 to 1.0 and, at the same time, the number of rule shortening operations is not too high which is essential for keeping the time needed to conduct computer experiments within acceptable bounds.

Now we present an algorithm of a stratifying classifier construction based on rule shortening (see Algorithm 3.1).

Let us notice that after the above algorithm completes its performance on the list L, there are eleven decision rule sets. The first classifier on this list contains the

Algorithm 3.1. Stratifying classifier construction

```
   Input: decision table $\mathbf{A} = (U, A, d)$ and concept $C \subseteq U$
   Output: classifier list $L$ representing a stratifying classifier
 1 begin
 2     Calculate decision rules for table $\mathbf{A}$, denoted by $RUL(\mathbf{A}) =$
       $RUL_1(\mathbf{A}) \cup RUL_2(\mathbf{A})$
 3     Create empty classifier list $L$
 4     for $a := 0.5$ to 0.9 with step 0.1 do
 5         Shorten rules $RUL_1(\mathbf{A})$ to the consistency coefficient $a$ and place
           the shortened decision rules in $RUL_1(\mathbf{A}, a)$
 6         $RUL := RUL_1(\mathbf{A}, a) \cup RUL_2(\mathbf{A})$
 7         Add $RUL$ to the end of the list $L$
 8     end
 9     Add $RUL(\mathbf{A})$ to the end of the list $L$
10     for $a := 0.9$ to 0.5 with step 0.1 do
11         Shorten rules $RUL_2(\mathbf{A})$ to the consistency coefficient $a$ and place
           the shortened decision rules the $RUL_2(\mathbf{A}, a)$
12         $RUL := RUL_1(\mathbf{A}) \cup RUL_2(\mathbf{A}, a)$
13         Add $RUL$ to the end of the list $L$
14     end
15     return $L$
16 end
```

Algorithm 3.2. Classification using the stratifying classifier

```
Input:
  1. classifier list $L$ representing a stratifying classifier,
  2. set of labels of layers $E = \{e_1, ..., e_{size(L)+1}\}$,
  3. tested object $u$
Output: The label of the layer to which the object $u$ is classified
begin
    for $i := size(L)$ down to 1 do
        Classify $u$ using the classifier $L[i]$
        if $u$ is classified by $L[i]$ to the concept $C$ then
            return $e_{i+1}$
        end
    end
    return $e_1$
end
```

most shortened rules classifying to C. That is why, if it classifies an object to C', the degree of certainty is the highest that this object belongs to concept C', whereas the last classifier on the list L, contains the most shortened rules classifying to C'. That is why the classification of an object to the concept C using this classifier gives us the highest degree of certainty of that the object really belongs to C.

The time complexity of Algorithm 3.1 depends on the time complexity of the chosen algorithm of decision rules computing and on the algorithm of approximate rules synthesis (see Section 2.5).

On the basis of the classifier constructed according to Algorithm 3.1, the tested object is classified to a specific layer with the help of successive classifiers starting from the last to the first one, and if the object is classified by the i-th classifier to C, then we learn about membership of the object under testing to the $(i+1)$-th layer of C. However, if the object is not classified to C by any classifier, we learn about membership of the tested object to the first layer (layer number 1), that is, to the complement of concept C. We present a detailed algorithm for classification of the object using the stratifying classifier (see Algorithm 3.2).

Let us notice that if the size of the list L is equal to 11 (generated by Algorithm 3.1), the above classifier classifies objects to 12 concept layers where the number 12 layer is the layer of objects with the highest degree of certainty of membership to the concept and the layer number 1 is the layer with the lowest degree of certainty of membership to this concept.

4 General Methodology of Complex Concept Approximation

Many real-life problems may be modeled with the help of the so-called *complex dynamical systems* (see, *e.g.*, [92, 93, 94, 95, 96, 97]) or, putting it in an other way, *autonomous multiagent systems* (see, *e.g.*, [98, 101]) or *swarm systems* (see, *e.g.*, [104]). These are sets consisting of complex objects which are characterized by the constant change of parameters of their components over time, numerous relationships among the objects, the possibility of cooperation/competition among the objects and the ability of objects to perform more or less complicated actions. Examples of systems of these kind are: *traffic*, *a patient observed during treatment*, *a team of robots during performing some task*. The description of the dynamics of such a system is often impossible using purely classical analytical methods, and the description itself contains many vague concepts. For instance, in order to monitor complex dynamical systems effectively, complex spatio-temporal concepts are used very often, concerning dynamic properties of complex objects occurring in these systems. These concepts are expressed in natural language on a much higher level of abstraction than the so-called sensor data, which have mostly been applied to approximation of concepts so far. Examples of such concepts are *safe car driving*, *safe overtaking*, *patient's behavior when faced with a life threat*, *ineffective behavior of robot team*.

Much attention has been devoted to spatio-temporal exploration methods in literature (see, *e.g.*, [63, 64]). The current experience indicates more and more that approximation of such concepts requires a support of knowledge of the domain to which the approximated terms are applied, i.e., the *domain knowledge*. It usually means the knowledge about concepts occurring in a given domain and various relations among these concepts. This knowledge exceeds significantly the knowledge gathered in data sets; it is often represented in a natural language, and it is usually obtained in a dialogue with an expert from a given domain (see, *e.g.*, [41, 42, 43, 44, 45, 46, 52, 269]). One of the methods of representing this knowledge is recording it in the form of the so-called *concept ontology*. The concept ontology is usually understood as a finite set of concepts creating a hierarchy and relationships among these concepts which connect concepts from different hierarchical levels (see next section). In this subsection, we present a general methodology of approximating complex spatio-temporal concepts on the basis of experimental data and a domain knowledge represented mainly by a concept ontology.

4.1 Ontology as a Representation of Domain Knowledge

The word *ontology* was originally used by philosophers to describe a branch of metaphysics concerned with the nature and relations of being (see, *e.g.*, [270]). However, the definition of ontology itself has been a matter of dispute for a long time, and controversies concern mainly the thematic scope to be embraced by this branch. Discussions on the subject of ontology definition appear in the works of Gottfried Leibniz, Immanuel Kant, Bernard Bolzano, Franz Brentano, or Kazimierz Twardowski (see, *e.g.*, [271]). Most of them treat ontology as a field of science concerning types and structures of objects, properties, events, processes, relations, and reality domains (see, *e.g.*, [106]). Therefore, ontology is neither a science concerning functioning of the world nor the ways a human being perceives it. It poses questions: *How do we classify everything?*, *What* *classes of beings are inevitable for describing and concluding on the subject of ongoing processes?*, *What classes of being enable to conclude about the truth?*, *What classes of being enable to conclude about the future?* (see, *e.g.*, [106, 270]).

Ontology in Informatics. The term ontology appeared in the information technology context at the end of the sixties of the last century as a specific way of knowledge formalization, mainly in the context of database development and artificial intelligence (see, *e.g.*, [53, 272]). The growth in popularity of database systems caused avalanche increase of their capacity. The data size, multitude of tools used both for storing and introducing, or transferring data caused that databases became difficult in managing and communication with the outside world. Database schemes are determined to high extent not only by the requirements on an application or database theory but also by cultural conditions, knowledge, and the vocabulary used by designers. As the result, the same class of objects may possess different sets of attributes in various schemes termed differently. These attribute sets are identical terms but often describe completely different things. A solution to this problem are supposed to be ontologies which can be treated as tools for establishing standards of database scheme creation.

The second pillar of ontology development is artificial intelligence (AI), mainly because of the view according to which making conclusions requires knowledge resources concerning the outside world, and ontology is a way of formalizing and representing such knowledge (see, *e.g.*, [7, 273]).

It is worth noticing that, in the recent years, one of the main applications of ontologies has been their use for an intelligent search of information on the Internet (see, *e.g.*, [53] and [54] for more details).

Definition of Ontology. Philosophically as well as in information technology, there is a lack of agreement if it comes to the definition of ontology. Let us now consider three definitions of ontology, well-known from literature. Guarino states (see [53]) that in the most prevalent use of this term, an ontology refers to *an engineering artifact, constituted by a specific vocabulary used to describe a certain reality (or some part of reality), plus a set of explicit assumptions regarding the intended meaning of vocabulary words.* In this approach, an ontology describes a hierarchy of concepts related by relationships, whereas in more sophisticated cases, suitable axioms are added to express other relationships among concepts and to constrain the interpretation of those concepts.

Another well-known definition of ontology has been proposed by Gruber (see [105]). He defines an ontology as *an explicit specification of a conceptualization.* He explains that for AI systems, what *exists* is that which can be represented. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the *universe of discourse.* This set of objects and the describable relationships among them are reflected in the representational vocabulary with which a knowledge-based program represents knowledge. Thus, according to Gruber, in the the context of AI, we can describe the ontology of a knowledge-based program by defining a set of representational terms. In such an ontology, definitions associate the names of entities in the universe of discourse (*e.g.*, classes, relations, functions, or other objects) with human-readable text describing what the names mean, and formal axioms that constrain the interpretation and the well-formed use of these terms.

Finally, we present a view of ontology recommended by *the World Wide Web Consortium* (W3C) (see [107]). W3C explains that *an ontology defines the terms used to describe and represent an area of knowledge.* Ontologies are used by people, databases, and applications that need to share domain information (a domain is just a specific subject area or area of knowledge such as medicine, tool manufacturing, real estate, automobile repair, financial management, etc.). Ontologies include computer-usable definitions of basic concepts in the domain and the relationships among them. They encode knowledge in a domain and also knowledge that spans domains. In this way, they make that knowledge reusable.

Structure of Ontology. Concept ontologies share many structured similarities, regardless of the language in which they are expressed. However, most ontologies describe individuals (objects, instances), concepts (classes), attributes (properties), and relations (see, *e.g.*, [53, 54, 105, 107]).

Individuals (objects, instances) are the basic, "ground level" components of an ontology. They may include concrete objects such as people, animals, tables, automobiles, and planets, as well as abstract individuals such as numbers and words.

Concepts (classes) are abstract groups, sets, or collections of objects. They may contain individuals or other concepts. Some examples of concepts are vehicle (the class of all vehicles), patient (the class of all patients), influenza (the class of all patients suffering from influenza), player (the class of players), team (the class of all players from some team).

Objects belonging to concepts in an ontology can be described by assigning attributes to them. Each attribute has at least a name and a value, and is used to store information that is specific to the object the attribute is attached to. For example, an object from the concept *participant* (see ontology from Fig. 5[2]) has attributes such as first name, last name, address, affiliation. If you did not define attributes for concepts, you would have either a taxonomy (if concept relationships are described) or a controlled vocabulary. These are useful, but are not considered true otologies.

There are three following types of relations between concepts from ontology: *a subsumption relation* (written as *is-a* relation), a *meronymy relation* (written as *part-of* relation), and a *domain-specific* relation.

The first type of relations is the *subsumption relation* (written as *is-a*). If a class A subsumes a class B, then any member of the class A *is-a* member of the class B. For example, the class *author* subsumes the class *participant.* It means that anything that is a member of the class *author* is a member of the class *Participant* (see ontology from Fig. 5). Where A subsumes B, A is called the *superclass*, whereas B is the *subclass.* The subsumption relation is very similar to the notion of *inheritance*, well-known from the *object-oriented programming*

[2] This example has been inspired by Jarrar (see [54]).

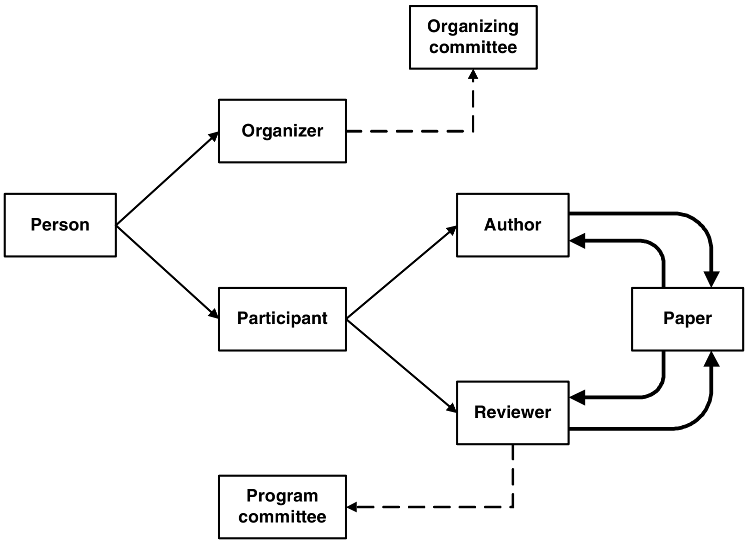


Fig. 5. The graph of a simple ontology

(see, *e.g.*, [274, 275]). Such relation can be used to create a hierarchy of concepts, typically with a maximally general concept like *person* at the top, and more specific concepts like *author* or *reviewer* at the bottom. The hierarchy of concepts is usually visualized by *a graph of ontology* (see Fig. 5) where any subsumption relation is represented by a thin solid line with an arrow in the direction from the superclass to the subclass.

Another common type of relations is the *meronymy relation* (written as *part-of*) that represents how objects combine together to form composite objects. For example, in the ontology from Fig. 5, we would say that any reviewer is-part-of the *program committee*. Any meronymy relation is represented graphically by a broken line with an arrow in the direction from the part to the composite object (see Fig. 5). From the technical point of view this type of relation between ontology terms is represented with the help of object attributes belonging to concepts. It is done in such a way that the value of an attribute of an object u, which is to be a part of some object u' belonging to different concept, informs about u'.

Apart from the standard *is-a* and *part-of* relations, ontologies often include additional types of relations that further refine the semantics modeled by the ontologies. These relations are often *domain-specific* and are used to answer particular types of questions. For example, in the domain of conferences, we might define a *written-by* relation between concepts *paper* and *author* which tells us who is the author of a paper. In the domain of conferences, we define also a *writes* relation between concepts *author* and *paper* which tells us which paper has been written by each author. Any domain-specific relation is represented by a thick solid line with an arrow. From the technical point of view this type of relations between ontology concepts is also represented with the help of object attributes belonging to the concepts.

In this paper, we use many ontologies, constructed on the basis of domain knowledge concerning the analyzed data sets, to approximate complex concepts. In these ontologies there occur all types of relations mentioned above. However, the relations of individual types do not occur in these ontologies simultaneously but in each of them there occurs only one type of relation. The reason for this is the fact that individual relation types serve us to approximate different types of complex concepts. For example, relations of the type *is-a* occur in ontology from Fig. 6 which is an example of an ontology used to approximate spatial concepts (see Section 5). Ontologies showing dependencies between temporal concepts for structured objects and temporal concepts for constituent parts of these objects (used to approximate temporal concepts for structured objects) are examples of ontologies in which there occur relations of type *part-of* (see Section 6). On the other hand, *domain-specific* relations occur in numerous examples of behavior graphs presented in Section 6 and are used to approximate behavioral patterns. The planning graphs presented in Section 7 are also examples of ontologies in which there occur *domain-specific* relations. Incidentally, planning graphs are, in a way, ontologies even more complex than the mentioned above, because two types of concepts occur in them simultaneously. Namely, there occur concepts representing states of complex objects and concepts representing actions performed on complex objects.

Obviously, there are many ways of linking the ontologies mentioned above provided they concern the same domain. For example, an ontology describing the behavior graph of a group of vehicles may be linked with ontologies describing dependencies of temporal concepts for such groups of vehicles and temporal concepts describing behavior of individual vehicles or changes of relationships among these vehicles. Then, in such an ontology, there would occur relations of two types simultaneously, that is, *domain-specific* and *part-of* relations. Although these types of linking different ontologies are not essential for complex concept approximation methods presented in this paper, they cause a significant increase of complexity of the ontologies examined.

General Recommendations Concerning Building of an Ontology. Currently there are many papers which describe various designer groups' experience obtained in the process of ontology construction (see, *e.g.*, [276]). Although they do not constitute formal frames enabling to create an integral methodology yet, general recommendations how to create an ontology may be formed on their basis. Each project connected with an ontology creation has the following phases:

- Motivation for creating an ontology.
- Definition of the ontology range.
- Ontology building.
  - Building of a lexicon.
  - Identification of concepts.

  - Building of the concept structure.
  - Modeling relations in ontology.
- The evaluation of the ontology obtained.
- Ontology implementation.

Motivation for creating an ontology is an initial process resulting from arising inside a certain organization, a need to change the existing ontology or to create a new one. Extremely crucial for the whole further process is, at this stage, clarity of the aim for which the ontology is built. It is the moment when potential sources of knowledge needed for the ontology construction should be defined. They are usually sources which may be divided into two groups those requiring human engagement (*e.g.*, interviews, discussions) and those in which a human does not appear as a knowledge source (*e.g.*, documents, dictionaries and publications from the modeled domain, intranet and Internet, and other ontologies).

By *the ontology range* we understand this part of the real world which should be included into the model under creation in the form of concepts and relations among them. One of the easier, and at the same time very effective, ways to determine the ontology range accurately is using the so-called "competency questions" (see, *e.g.*, [277]). The starting point for this method is defining a list of questions to which the database built on the basis of the ontology under construction should give an answer.

Having the range defined, the process of *ontology building* should be started. The first step in ontology building is defining a list of expressions, phrases, and terms crucial for a given domain and a specific context of application. A *lexicon* should be composed that is a dictionary containing terms used by the ontology as well as their definitions, from the list.

The lexicon is a starting point for the most difficult stage in the ontology building, that is, construction of concepts (classes) of the ontology and relations among these concepts. It should be remembered that it is not possible to perform these two activities one after the other. They have to be performed in parallel. We should bear in mind that each relation is also a concept. Thus, finding the answer to the question *What should constitute a concept and what should constitute a relation?* is not easy and depends on the target application and, often, the designer's experience.

If it comes to building hierarchical classes, three approaches to building such a hierarchy are given in the paper [278]:

1. *Top-down.* We start with a concept superior to all concepts included in the knowledge base and we come to the next levels of inferior concepts by applying atomization.
2. *Bottom-up.* We start with the most inferior concept contained in the knowledge base and we come to the concepts on higher levels of hierarchy by applying generalization.
3. *Middle-out.* We start with concepts which are the most crucial in terms of the project and we perform atomization or generalization when needed.

In order to evaluate the obtained ontology it should be checked if the ontology possesses the following qualities ([277]):

– *Consistency.* The ontology is consistently integral, that is, contradictory conclusions cannot be drawn from it.
– *Completeness.* The ontology is complete if all expected elements are included in the model (concepts, relations, etc.).
– *Conciseness.* All information gathered in the ontology is concise and accurate.
– The possibility of answering the *"competency questions"* posed previously.

Summing up, an ontology building is a laborious process requiring a huge amount of knowledge concerning the modeling process itself, the tools used, and the domain being modeled.

Ontology Applications. Practical ontology applications relate to the so-called *general ontologies* which have a rather general character and may be applied in a knowledge base building from different domains and *domain ontologies* meaning ontologies describing knowledge about a specific domain or a specific fragment of the real world. Many such ontologies have been worked out and they are often available on the Internet. They are, *e.g.*, Dublin Core (see [279]), GFO (General Formal Ontology [280]), OpenCyc/ResearchCyc (see [281]), SUMO (Suggested Upper Merged Ontology [282]), WordNet (see [283]), DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering [284]) and others.

Generally, ontologies are applied when the semantics of the data gathered is crucial. It turns out that such a situation takes place quite often, particularly when intelligent methods of data analysis are supposed to act effectively. That is why ontologies more and more are useful in information technology projects. Some examples of applications of ontologies are e-commerce, bioinformatics, geographical, information systems, regulatory and legal information systems, digital libraries, e-learning, agent technology, database design and integration, software engineering natural language processing, information access and retrieval, the Semantic Web, Web services, medicine (see, *e.g.*, [53] and [54] for more details).

Computer Systems for Creating and Using Ontologies. There is a series of formal languages to represent ontologies. These are such languages as *Web Ontology Language* (OWL [107]), *Resource Description Framework* (RDF [285]), *Ontology Inference Layer* (OIL [286]), *DARPA Agent Markup Language* (DAML [287]), CycL (see [288]), etc. However, the most dynamically developed one is OWL which came to the existence as an improvement of the DAML, OIL and RDF languages.

There are also many computer systems for creating and using ontologies. They are, *e.g.*, Cyc (see [288]), OpenCyc (see [289]), Protege (see [290]), OntoStudio (previously OntoEdit [291]), Ontolingua (see [292]), Chimaer (see [293]), OilEd (see [294]), and others. Within these systems, the ontology is usually created using convenient graphical tools which make it possible to enter all the elements of ontology as well as their further edition and visualization.

Ontological systems very often possess mechanisms of concluding on the basis of ontology constructed. These mechanisms work in such a way that after creating an ontology the system may be asked quite complex questions. They concern the existence of an instance of a concept which satisfies certain logical conditions, defined using concepts, attributes, and relations occurring in the ontology. For instance, in the ontology in Fig. 5, we could pose the following questions:

– *Who is the author of a given paper?*
– *Which papers have been reviewed by a given reviewer?*
– *Which persons belong to the programming committee?*

From the technical point of view, information searching based on ontology is performed with the help of questions formed in a formal language used to represent ontology or its special extension. For instance, the language RDQL (*RDF Data Query Language* [295]) is a question language similar to the language SQL extending the RDF language. Usually, the ontological systems also enable to form questions using graphical interface (see, e.g, [290, 291]).

4.2 Motivations for Approximation of Concepts and Relations from Ontology

In current systems operating on the basis of ontology it is assumed that we possess complete information about concepts, that is, for each concept all objects belonging to this concept are known by us. This assumption causes that, in order to examine the membership of an object to the concept, it is enough to check if this object occurs as an instance of this concept or not. Meantime, in practical applications we often possess only incomplete information about concepts, that is, for each concept, certain sets of objects constituting examples and counterexamples, respectively are given. It causes the necessity of approximating concepts with the help of classifiers. For instance, using the ontology in Fig. 6 which concerns safe vehicle driving on a road, it cannot be assumed that all concept instances of this ontology are available. For example, for the concept *safe driving*, it cannot be assumed that the information about all possible cars driving safely is available. That is why for such a concept, a classifier is constructed which is expected to be able to classify examples of vehicles into those belonging and those which do not belong to the concept.

Apart from that, the relations between concepts, defined in current systems based on ontology, are usually precise (exact, crisp). For example, for the relation *is-a* in ontology from Fig. 5, if the relation between concepts *author* and *participant* is to be precise (exact, crisp), then each author of a paper at a conference is a participant of this conference. In practice, however, it does not always have to be that way. It is possible that some authors of papers are not conference participants, particularly in the case of articles having many coauthors. So, a relation between concepts can be imprecise (inexact, vague). Besides, on the grounds of classical systems based on ontology, when we possess complete information about concepts, the problem of vagueness of the above relation may be solved by adding to the ontology an additional concept representing these authors who are not conference participants and binding this new concept with the concept *person* by the *is-a* relation. However, in practical applications, when the available information about concepts is not complete, we are even not able to

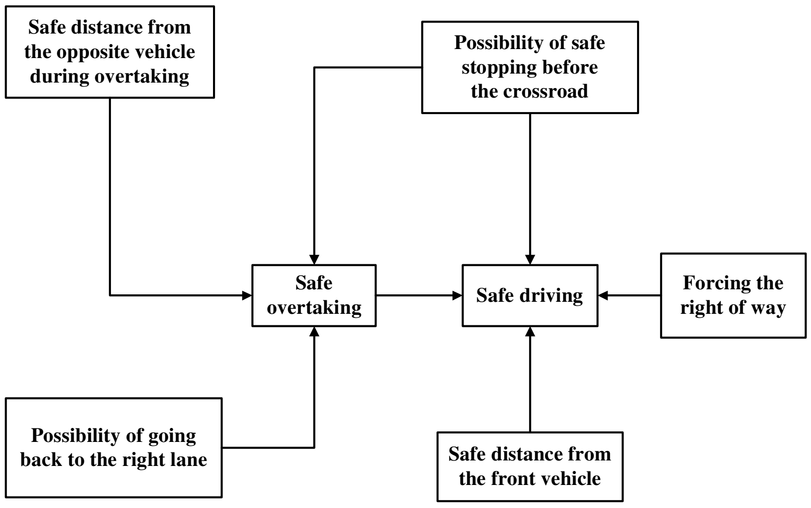

Safe distance from the opposite vehicle during overtaking
Possibility of safe stopping before the crossroad
Safe overtaking
Safe driving
Forcing the right of way
Possibility of going back to the right lane
Safe distance from the front vehicle

Fig. 6. An ontology for safe driving

check whether the relations under consideration are precise (exact, crisp). That is why relations among concepts also require approximation.

In approximation of concepts occurring in ontology, there often appears the following problem. In practical applications, usually is the so-called sensor data available only (that is, data obtained by measurement using sensors, thus obtained on a low level of abstraction). For example, by observing a situation on a road, i.e., such data as speed, acceleration, location, the current driving lane, may be obtained. Meanwhile, some concepts occurring in ontology are so complex that they are separated by a considerable semantical distance from the sensor data, i.e., they are defined and interpreted on very different levels of abstraction. Hence, approximation of such concepts using sensor data does not lead to classifiers of satisfying quality (see, *e.g.*, [42, 44, 45, 46, 48]). For instance, in ontology from Fig. 6, such a complex concept is without a doubt the concept *safe driving* because it is not possible to directly determine whether a given vehicle goes safely on the basis of simple sensor data only.

If, however, apart from complex concepts there are simple concepts in ontology, that is, those which may be approximated using sensor data, and they are directly or indirectly linked by relations to complex concepts, then appears a need to use the knowledge about the concepts and relations among them to approximate complex concepts more effectively. For example, in order to determine if a given vehicle drives safely, other concepts from ontology from Fig. 6, linked by relations to the concept *safe driving*, may be used. For example, one of such concepts is the *possibility of safe stopping before the crossroad.*

The aim of this paper is to present set of methods for approximating complex spatio-temporal concepts and relations among them assuming that the

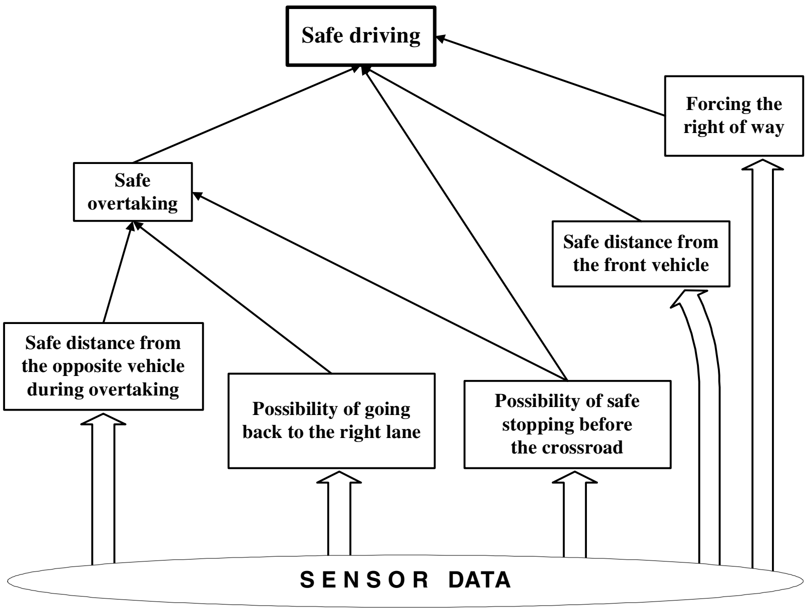

Safe driving
Forcing the right of way
Safe overtaking
Safe distance from the front vehicle
Safe distance from the opposite vehicle during overtaking
Possibility of going back to the right lane
Possibility of safe stopping before the crossroad
SENSOR DATA

Fig. 7. The ontology for safe driving revisited

information about concepts and relations is given in the form of ontology. To meet these needs, by ontology we understand a finite set of concepts creating a hierarchy and relations among these concepts which link concepts from different levels of the hierarchy. At the same time, on top of this hierarchy there is always the most complex concept whose approximation we are interested in aiming at practical applications. For example, ontology from Fig. 6 may be presented hierarchically as in Fig. 7. At the same time, we assume that the ontology specification contains incomplete information about concepts and relations occurring in ontology, particularly for each concept, sets of objects constituting examples and counterexamples for these concepts are given. Additionally, for concepts from the lowest hierarchical level (sensor level) it is assumed that there are also *sensor attributes* available which enable to approximate these concepts on the basis of the examples and counterexamples given. This fact is marked in Fig. 7 by block arrows.

4.3 Unstructured, Structured, and Complex Objects

Every concept mentioned in this paper is understood as a subset of a certain set called the universe. Elements of the universe are called objects and they are interpreted as states, incidents, vehicles, processes, patients, illnesses and sets or sequences of entities mentioned previously. If such objects come from the real-life world, then their perception takes place by detecting their structure. Discovery of relevant object structure for particular tasks is a complex problem strongly related to perception, that is usually understood as the process of acquiring, interpreting, selecting, and organizing sensory information (see, *e.g.*, [45, 86, 145, 146, 147, 148, 149, 150, 151, 152, 296, 297, 298, 299]). Many interdisciplinary research has been conducted in this scope in the overlapping areas of such fields as cognitive science, psychology and neuroscience, pattern recognition (see, *e.g.*, [26, 27, 35, 300, 301, 302, 303]).

Structure of objects is used to define compound patterns over objects with the simple or structured structures. The construction of such compound patterns may be hierarchical. We search for patterns relevant for approximation of some complex concepts. Notice, that together with the granularity of patterns one should consider the computational complexity of satisfiability testing for such patterns.

The structure of the perceived objects may be more or less complex, because the objects may differ in complexity. It means both the degree of spatial as well as the spatio-temporal complexity. When speaking about spatial complexity we mean not only the fact that the objects differ in the features such as location, size, shape, color, weight, but also that objects may consist of parts related with each other in terms of dependencies (*e.g.*, one may examine objects which are object groups in the traffic). However, the spatio-temporal complexity results from the fact that the perception of objects may be extended over time (*e.g.*, one may examine objects which are single vehicles observed at a single time point and objects which are also single vehicles, but they are observed over a certain period of time). Both of these aspects of object complexity may cumulate which additionally increases the diversity of appearing objects (*e.g.*, objects which are vehicle groups observed over a certain period of time are more complex than both the objects which are vehicle groups observed at a single time point and the objects which are single vehicles observed over a certain period of time).

However, in practice the perception of objects always takes place on an established level of perceiving detail. This means that depending on the needs, during perceiving objects only such details concerning their structure are taken into account that are necessary to conduct effective reasoning about the objects being perceived. For example, if we want to identify vehicles driven dangerously on the road, then we are not interested in the internal construction of each vehicle but rather the behavior of each vehicle as a certain whole. Hence, in the paper, we examine objects of two types. The first type of objects are *unstructured* objects, meaning those which may be treated as indivisible wholes. We deal with this type of objects when we analyze *patients*, *bank clients* or *vehicles*, using their parameters observed at the single time point.

The second type of objects which occurs in practical applications are *structured* objects which cannot be treated as indivisible wholes and are often registered during some period. Examples of this type of objects may be *a group of vehicles driving on a highway*, *a set of illnesses occurring in a patient*, *a robot team performing a task.*

In terms of spatiality, structured objects often consist of disjunctive parts which are objects of uniform structure connected with dependencies. However, generally, the construction of structured objects is hierarchical, that is, their parts may also be structured objects. Additionally, a great spatial complexity of structured objects causes that conducting effective reasoning about these objects usually requires their observation over a certain period of time. Thus, the hierarchy of such objects' structure may concern not only their spatial, but also spatio-temporal structure. For example, to observe simple behaviors of a single vehicle (*e.g.*, speed increase, a slight turn towards the left lane) it is sufficient to observe the vehicle over a short period of time, whereas to recognize more complex behaviors of a single vehicle (*e.g.*, acceleration, changing lanes from right to the left one), the vehicle should be observed for a longer period of time, at the same time a repeated observation of the above mentioned simple behaviors may be extremely helpful here (*e.g.*, if over a certain period the vehicle increased speed repeatedly, it means that this vehicle probably accelerates). Finally, behavior observation of a vehicle group requires its observation for an even longer period of time. It happens that way because the behavior of a vehicle group is usually the aggregation or consequence of vehicle behaviors which belong to the group (*e.g.*, observation of an overtaking maneuver of one vehicle by another requires following specific behaviors of both the overtaking and overtaken vehicle for a certain period of time).

Obviously, each of structured objects usually may be treated as an unstructured object. If we treat any object as an unstructured object at a given moment, it means that its internal structure does not interest us from the point of view of the decision problems considered. On the other hand, it is extremely difficult to find real-life unstructured objects, that is, objects without parts. In the real-life world, almost every object has some kind of internal structure and consists of certain spatial, temporal or spatio-temporal parts. Particularly, objects which are examples and counterexamples of complex concepts (both spatial and spatio-temporal), being more or less semantically distant from sensor data, have a complex structure. Therefore, one can say that they are *complex objects*. That is why the division of complex objects into unstructured and structured ones is of a symbolic character only and depends on the interpretation of these objects. If we are interested in their internal structure, then we treat them as structured objects; otherwise we treat them as unstructured ones.

4.4 Representation of Complex Object Collections

If complex objects are gathered into a collection, then in order to represent the available information about these objects, one may use information systems. Below, we present an example of such an information system whose objects are vehicles and attributes describe the parameters of the vehicle recorded at a given time point.

Example 1. Let us consider an information system $\mathbf{A} = (U, A)$ such that $A = \{x, y, l, v, t, id\}$. Each object of this system represents a condition of a considered vehicle at one time moment. The attributes x and y provide the current location of a vehicle, the l and v attributes provide us with current traffic lane on which the vehicle is and the current vehicle speed respectively. The attribute t represents time in a number of seconds which has passed since the first observation of the vehicle (V_t is a subset of the set of positive integer numbers). The attribute id provides identifiers of vehicles. $\square$

The second, extremely crucial, example of the information system used in this paper is an information system whose objects represent patient conditions at different time points.

Example 2. Let us consider an information system $\mathbf{A} = (U, A)$ such that $U = \{u_1, ..., u_n\}$ and $A = \{a_1, ..., a_m, a_t, a_{id}\}$. Each object of this system represents medical parameters of a certain patient during one day of his/her hospitalization. Attributes $a_1, ..., a_m$ describe medical parameters of the patient (examination results, diagnoses, treatments, medications, etc.), whereas the attribute a_t represents time in a number of days which has passed since the first observation of the patient (V_{a_t} is a subset of the set of positive integer numbers). Finally, the attribute a_{id} provides identifiers of patients. $\square$

Similarly to the two examples above, the attributes of complex objects may be based on sensor data. However, in a general case the properties of complex objects may be defined in languages which are defined specifically for a given purpose (see Section 4.7).

4.5 Relational Structures

As we have written before, structured objects consist of parts which are structured objects of lesser complexity (hierarchical structure) or unstructured objects connected by dependencies. Additionally, a great spatial complexity of structured objects causes that conducting effective conclusions about these objects usually requires their observation for a certain period of time. Hence, there is a need to follow the spatio-temporal dependencies between parts of complex objects. Therefore, the effective description of the structure of objects requires not only providing spatial properties of individual parts of these objects, but also describing the spatio-temporal relations between the parts of these objects. Therefore, in order to describe the structure of complex objects and relations between complex objects in this paper we will use relational structures (see, *e.g.*, [5, 89]).

In order to define the relational structure using language and semantics of *first-order logic* we assume that a set of relation symbols $REL = \{R_i : i \in I\}$ and function symbols $FUN = \{f_j : j \in J\}$ are given, where I, J are some finite sets (see, *e.g.*, [89]). For any functional or relational symbol there is assigned a natural number called the *arity* of the symbol. Functional symbols and relations of arity 0 are called *constants*. The set of constants is denoted by $CONST$. Symbols of arity 1 are called *unary* and of arity 2 are called *binary*. In the case of binary relational or functional symbols we usually use traditional infix notation. For instance we write $x \leq y$ rather than $\leq (x, y)$. The set of functional and relational symbols together with their arities is called the *signature*. The interpretation of a functional symbol f_i (a relational symbol R_i) over the set A is a function (a relation) defined over the set A and denoted by f_i^A (R_i^A). The number of arguments of a function f_i^A (a relation R_i^A) is equal the arity of f_i (R_i). Now, we can define the relational structure of a given signature (see, *e.g.*, [5, 89]).

Definition 2 (*A relational structure of a given signature*). *Let $\Sigma = REL \cup FUN$ be a signature, where $REL = \{R_i : i \in I\}$ is a set of relation symbols and $FUN = \{f_j : j \in J\}$ is a set of function symbols, where I, J are some finite sets.*

1. *A relational structure of signature Σ is a triple $(D, \mathbf{R}, \mathbf{F})$ where*
   - *D is a non-empty finite set called the domain of the relational structure,*
   - *$\mathbf{R} = \{R_1^D, ..., R_k^D\}$ is a finite (possibly empty) family of relations defined over D such that R_i^D corresponds to symbol $R_i \in REL$ and $R_i^D \subseteq D^{n_i}$ where $0 < n_i \leq card(D)$ and n_i is the arity of R_i, for $i = 1, ..., k$,*
   - *$\mathbf{F} = \{f_1^D, ..., f_l^D\}$ is finite (possibly empty) family of functions such that f_j^D corresponds to symbol $f_j \in FUN$ and $f_j^D : D^{m_j} \longrightarrow D$ where $0 \leq m_j \leq card(D)$ and m_j is the arity of f_j, for $j = 1, ..., l$.*
2. *If for any $f \in \mathbf{F}$, $f : D^0 \longrightarrow D$, then we call such a function constant and we identify it with one element of the set D, corresponding to f.*
3. *If $(D, \mathbf{R}, \mathbf{F})$ is a relational structure and F is empty, then such relational structure is called pure relational structure and is denoted by $(D, \mathbf{R})$.*

A classical example of a relational structure is a set of real numbers with operations of addition and multiplications and ordering relation.

A typical example of a pure relational structure is a directed graph whose domain is set of graph nodes and the family of relations consists of one relation described by a set of graph edges.

The example below illustrates how relational structures may be used to describe the spatial structure of a complex object.

Example 3. Let us examine the complex object which is perceived as an image in Fig. 8. In this image one may notice a group of six cars: A, B, C, D, E, F. In order to define the spatial structure of this car group, the most crucial thing is defining the location of cars towards each other and defining the diversity of the driving directions of individual cars. That is why the spatial structure of such a group may be described with the help of relational structure $(\mathcal{S}, \mathbf{R})$, where:

- $\mathcal{S} = \{A, B, C, D, E, F\}$,
- $\mathbf{R} = \{R_1, R_2, R_3, R_4\}$, where:
  - $\forall (X, Y) \in \mathcal{S} \times \mathcal{S} : (X, Y) \in R_1$ iff X is driving directly before Y,
  - $\forall (X, Y) \in \mathcal{S} \times \mathcal{S} : (X, Y) \in R_2$ iff X is driving directly behind Y,
  - $\forall (X, Y) \in \mathcal{S} \times \mathcal{S} : (X, Y) \in R_3$ iff X is coming from the opposite direction in comparison with Y,
  - $\forall (X, Y) \in \mathcal{S} \times \mathcal{S} : (X, Y) \in R_4$ iff X is driving in the same direction as Y.

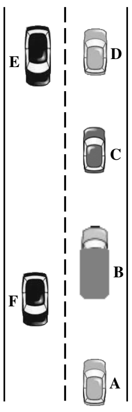


Fig. 8. An example of spatial complex object

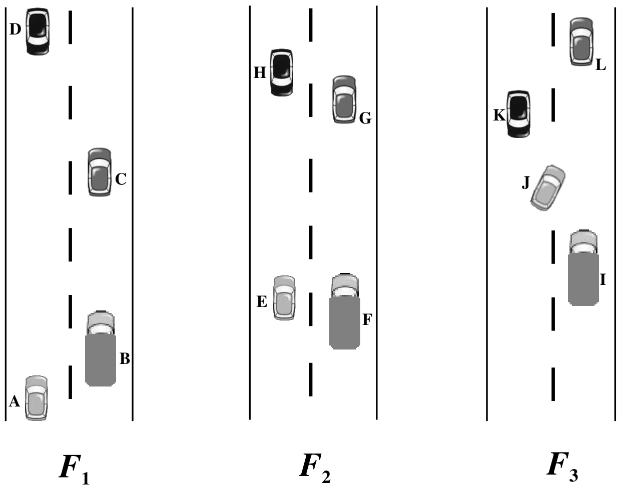


Fig. 9. An example of spatio-temporal complex object

For instance, it is easy to see that (B, A), (C, B), (D, C), $(F, E) \in R_1$, (A, B), (B, C), (C, D), $(E, F) \in R_2$, (E, C), (E, B), $(F, A) \in R_3$ and (A, C), (B, D), $(E, F) \in R_4$. □

Complex objects may also have spatio-temporal structure. The example below shows this type of a structured object.

Example 4. Let us examine the complex object which is represented with the help of three images F_1, F_2 and F_3 recorded at three consecutive time points (see Fig. 9). In image F_1 one may notice cars A, B, C and D, whereas in image

F_2 we see cars E, F, G and H. Finally, in image F_3 we see cars I, J, K and L (see Fig. 9). It is easy to notice that pictures F_1, F_2 and F_3 may be treated as three frames chosen from a certain film made *e.g.*, from an unmanned helicopter conducting a road observation, and at the same time each consecutive frame is distant in time from the previous one by about one second. Therefore, in all these pictures we see the same four cars, at the same time the first car is perceived as car A, E or J, the second car is perceived as car B, F or I, the third car is perceived as car C, G or L and the fourth car is perceived as car D, H or K. The spatial structure of complex object $\mathcal{ST} = \{A, B, C, D, E, F, G, H, I, J\}$ may be described with the help of relational structure similar to the one in Example 3. However, object $\mathcal{ST}$ has spatio-temporal structure which should be reflected in relational structure describing complex object $\mathcal{ST}$. That is why, to the relation family $\mathbf{R}$ from Example 3 we add relation R_t determined in the following way:

$$\forall (X,Y) \in \mathcal{ST} \times \mathcal{ST} : (X,Y) \in R_t \text{ iff } X \text{ represents the same vehicle as } Y \text{ and } X \text{ was recorded earlier than } Y.$$

For instance, it is easy to see that (A, E), $(H, K) \in R_t$, but (G, C), $(I, F) \notin R_t$ and (C, H), $(F, K) \notin R_t$.

Moreover, we modify the definition of the remaining relations from family $\mathbf{R}$:

- $\forall (X,Y) \in \mathcal{ST} \times \mathcal{ST} : (X,Y) \in R_1$ iff X, Y were noticed in the same frame and X is going directly before Y,
- $\forall (X,Y) \in \mathcal{ST} \times \mathcal{ST} : (X,Y) \in R_2$ iff X, Y were noticed in the same frame and X is driving directly behind Y,
- $\forall (X,Y) \in \mathcal{ST} \times \mathcal{ST} : (X,Y) \in R_3$ iff X, Y were noticed in the same frame and X is coming from the opposite direction in comparison with Y,
- $\forall (X,Y) \in \mathcal{ST} \times \mathcal{ST} : (X,Y) \in R_4$ iff X, Y were noticed in the same frame and X is driving in the same direction as Y.

□

If some set of complex objects is perceived as an unstructured object (its parts are not distinguished) and these objects belong to the object set of a certain information system, then a structure of such set of complex objects is described by relational structure, that we call *a trivial relational structure.*

Definition 3. *Let $\mathbf{A} = (U, A)$ be an information system. For any set of objects $U' \subseteq U$ we define a relational structure $(Dom, \mathbf{R}, \mathbf{F})$ such that $Dom = \{U'\}$, $\mathbf{R}$ and $\mathbf{F}$ are empty families. Such relational structure is called a trivial relational structure.*

The above trivial relational structures are used to approximate spatial concepts (see Section 5).

In each collection of complex objects there may occur relations between objects belonging to this collection. That is why each collection of complex objects may be treated as a complex object whose parts are objects belonging to the collection. Hence, the structure of complex object collection may be described using relational structure, where the domain elements of this structure are objects which belong to this collection (see Section 4.7).

4.6 Languages and Property Systems

In the paper, we use many special languages to define features of complex objects. Any language $\mathcal{L}$ is understood as a set of formulas over a given finite alphabet and is constructed in the following way.

1. First, we define an alphabet of $\mathcal{L}$, some atomic formulas and their semantics by means of some satisfiability relation $\models_{\mathcal{L}}$. The satisfiability relation is a binary relation in $X \times \mathcal{L}$, where X denotes a universe of elements (objects). We will write $x \models_{\mathcal{L}} \alpha$ to denote the fact that $\models_{\mathcal{L}}$ holds for the pair (x, α) consisting of the object x and the formula α.
2. Next, we extend, in the standard way, the satisfiability relation $\models_{\mathcal{L}}$ on Boolean combination of atomic formulas, i.e., on the least set of formulas including atomic formulas and closed with respect to the classical propositional connectives: disjunction ($\vee$), conjunction ($\wedge$), negation ($\neg$) using the following rules:
   (a) $x \models_{\mathcal{L}} (\alpha \vee \beta)$ iff $x \models_{\mathcal{L}} \alpha$ or $x \models_{\mathcal{L}} \beta$,
   (b) $x \models_{\mathcal{L}} (\alpha \wedge \beta)$ iff $x \models_{\mathcal{L}} \alpha$ and $x \models_{\mathcal{L}} \beta$,
   (c) $x \models_{\mathcal{L}} \neg(\alpha)$ iff $non(x \models_{\mathcal{L}} \alpha)$,
   where α, β are formulas, x is an object, and the symbol $\models_{\mathcal{L}}$ denotes the satisfiability relation of the defined language.
3. Finally, for any formula $\alpha \in \mathcal{L}$, the set $|\alpha|_{\mathcal{L}} = \{x \in X : x \models_{\mathcal{L}} \alpha\}$ can be constructed that is called the meaning (semantics) of α in $\mathcal{L}$.

Hence, in the sequel, in specifying languages and their semantics we will only define atomic formulas and their semantics assuming that the extension on Boolean combination is the standard one. Moreover, in definitions of alphabets over which languages are constructed we often omit listing parentheses assuming that the relevant parentheses are always included in alphabets.

Besides, in modeling complex objects we often use structures called *property systems.*

Definition 4 (*A property system*). *A property system is any triple $\mathcal{P} = (X, \mathcal{L}, \models)$, where X is a set of objects; $\mathcal{L}$ is a language over a given finite alphabet; and $\models \subseteq X \times \mathcal{L}$ is a satisfiability relation.*

We also use the following notation:

1. We write, if necessary, $X_{\mathcal{P}}$, $\mathcal{L}_{\mathcal{P}}$, $\models_{\mathcal{P}}$, instead of X, $\mathcal{L}$, and $\models$, respectively.
2. $|\alpha|_{\mathcal{P}} = \{x \in X : x \models_{\mathcal{P}} \alpha\}$ is the meaning (semantics) of α in $\mathcal{P}$.
3. By a_α for $\alpha \in \mathcal{L}_{\mathcal{P}}$ we denote a function (attribute) from $X_{\mathcal{P}}$ into $\{0, 1\}$ defined by $a_\alpha(x) = 1$ iff $x \models_{\mathcal{P}} \alpha$ for $x \in X_{\mathcal{P}}$.
4. Any property system $\mathcal{P}$ with a finite set of objects and a finite set of formulas defines an information system $\mathbf{A}_{\mathcal{P}} = (X_{\mathcal{P}}, A)$, where $A = \{a_\alpha\}_{\alpha \in \mathcal{L}}$.

It is worthwhile mentioning that the definition of any information system $\mathbf{A} = (U, A)$ constructed in hierarchical modeling should start from definition of the universe of objects of such an information system. For this purpose, we select a language in which a set U^* of complex objects is defined, where $U \subseteq U^*$. For specifying the universe of objects of $\mathbf{A}$, we construct some property system $\mathcal{Q}$ over the universe U^* of already constructed objects. The language $\mathcal{L}_\mathcal{Q}$ consists of formulas which are used for specifying properties of the already constructed objects from U^*. To define the universe of objects of $\mathbf{A}$, we select a formula α from $\mathcal{L}_\mathcal{Q}$. Such a formula is called *type* of the constructed information system $\mathbf{A}$. Now, we assume that the object x belongs to the universe of $\mathbf{A}$ iff x satisfies (in $\mathcal{Q}$) the formula α, i.e., $x \models_\mathcal{Q} \alpha$, where $x \in U^*$. Observe, that the universe of objects of $\mathbf{A}$ can be an extension of the set U because U is usually only a sample of possible objects of $\mathbf{A}$. Notice that the type α selected for a constructed information system defines a binary attribute a_α for this system. Certainly, this attribute can be used to define the universe of the information system $\mathbf{A}$ (see Section 4.7 for more details). Notice also that the property system $\mathcal{Q}$ is constructed using property systems and information systems used in modeling the lower level of concept hierarchy.

4.7 Basic Languages of Defining Features of Complex Objects

As we have written before, the perception of each complex object coming from the real-life world takes place by detecting its structure (see Section 4.3), whereas the features of a given complex object may be determined only by establishing the features of this structure. The structures of complex objects which are the result of perception of complex objects may be modeled with the help of relational structures (see Section 4.5). Therefore, by the features of complex objects represented with relational structures we will understand the features of these structures.

Each collection of complex objects K may be represented using an information system $\mathbf{A} = (U, A)$, where the object set U is equal to collection K and the attributes from set A describe the properties of complex objects from collection K and more precisely, the properties of relational structures representing individual objects from this collection.

In the simplest case, the attributes from set A may be sensor attributes, that is, they represent the readings of sensors recorded for objects from set U (see Example 1 and Example 2 from Section 4.4).

However, in the case of structured objects whose properties usually cannot be described with the help of sensor attributes, the attributes from set A may be defined with the use of the properties of these objects' parts, the relations between the parts and information about the hierarchy of parts expressed *e.g.*, with the help of concept ontology (see Section 4.10).

In practice, apart from the properties of complex objects described above and represented using the attributes from set A, other properties of complex objects are also possible which describe the properties of these objects on a slightly higher level of abstraction than the attributes from set A. These properties are usually defined by experts on the basis of domain knowledge and are often represented with the help of concepts, that is, attributes which have only two values. For the table in Example 1, *e.g.*, "safe driving"could be such a concept.

By adding such an attribute to the information system, which is usually called decision attribute or decision and marking it with d, we obtain decision table (U, A, d). However, effective approximation of a decision attribute d using attributes from set A usually requires defining new attributes which are often binary attributes representing concepts. Such concepts may be defined in an established language on the basis of attributes available in set A.

In this paper, such language is called *a language for defining features of complex objects*. In the simplest case such a language may be the language of mathematical formulas in which formulas enabling calculating the specific properties of a complex object are formed. For example, if the complex object is a certain subset of a set of rational numbers with simple addition and multiplication and the order relation, then the attributes of such a complex object may be: the minimal value, the maximum value or the arithmetic average over this set.

However, in many cases in order to define attributes of complex objects special languages should be defined. In this paper, to define a specific language defining complex object properties Tarski's approach is used which requires the language's alphabet, set of language formulas and language formula semantics (see, *e.g.*, [304] and Section 4.6).

For example, in order to define concepts describing new properties of objects from a given information system a well known language called *generalized descriptor language* may be used (see, *e.g.*, [16, 165]).

Definition 5 (*A generalized descriptor language***).** *Let $\mathbf{A} = (U, A)$ be an information system. A generalized descriptor language of information system $\mathbf{A}$ (denoted by $GDL(\mathbf{A})$ or GDL-language, when $\mathbf{A}$ is fixed) is defined in the following way:*

- *the set $AL_{GDL}(\mathbf{A}) = A \cup \bigcup_{a \in A} V_a \cup \{\neg, \vee, \wedge\}$ is an alphabet of the language $GDL(\mathbf{A})$,*
- *expressions of the form $(a \in V)$, where $a \in A$ and $V \subseteq V_a$ are atomic formulas of the language $GDL(\mathbf{A})$.*

Now, we determine the semantics of the language $GDL(\mathbf{A})$. The language $GDL(\mathbf{A})$ formulas may be treated as the descriptions of object occurring in system $\mathbf{A}$.

Definition 6. *Let $\mathbf{A} = (U, A)$ be an information system. The satisfiability of an atomic formula $\phi = (a \in V) \in GDL(\mathbf{A})$ by an object $u \in U$ from table $\mathbf{A}$ (denoted by $u \models_{GDL(\mathbf{A})} \phi$) is defined in the following way:*

$$u \models_{GDL(\mathbf{A})} (a \in V) \text{ iff } a(u) \in V.$$

We still need to answer the question of defining the atomic formulas (expressions of the form $a \in V$) belonging to the set of formulas of the above language.

In the case of symbolic attributes, in practical applications the formulas of the form $a \in V$ are usually defined using relations: "=" or "$\neq$" (*e.g.*, $a = v_a$ or $a \neq v_a$ for some symbolic attribute a such that $v_a \in V_a$). However, if the attribute a is a numeric one, then the correct atomic formulas may be $a < v_a$, $a \leq v_a$, $a > v_a$ or $a \geq v_a$. Atomic formulas may be also defined using intervals, for example: $a \in [v_1, v_2]$, $a \in (v_1, v_2]$, $a \in [v_1, v_2)$ or $a \in (v_1, v_2)$, where $v_1, v_2 \in V_a$.

We present a few examples of formulas of the language $GDL(\mathbf{A})$, where $\mathbf{A} = (U, A)$, $A = \{a_1, a_2, a_3\}$ and $v_1 \in V_{a_1}$, $v_2 \in V_{a_2}$ and $v_3, v_4 \in V_{a_3}$.

- $(a_1 = v_1) \wedge (a_2 \neq v_2) \wedge (a_3 \in [v_3, v_4))$,
- $(a_1 \neq v_1) \vee (a_2 = v_2)$,
- $((a_1 = v_1) \vee (a_2 = v_2)) \wedge (a_3 > v_3)$,
- $\neg((a_1 = v_1) \wedge (a_3 \leq v_3)) \vee ((a_2 \neq v_2) \wedge (a_3 \in (v_3, v_4]))$.

Another example of a language defining complex object properties may be *a neighborhood language*. In order to define the neighborhood language a dissimilarity function of pairs of objects of the information system is needed.

Definition 7. *Let $\mathbf{A} = (U, A)$ be an information system.*

1. *We call a function $DISM_{\mathbf{A}} : U \times U \longrightarrow [0, 1]$ the dissimilarity function of pairs of objects in the information system $\mathbf{A}$, if the following conditions are satisfied:*
   (a) *for any pair $(u_1, u_2) \in U \times U$:*
   $$DISM_{\mathbf{A}}(u_1, u_2) = 0 \Leftrightarrow \forall\ a \in A :\ a(u_1) = a(u_2),$$
   (b) *for any pair $(u_1, u_2) \in U \times U$: $DISM_{\mathbf{A}}(u_1, u_2) = DISM_{\mathbf{A}}(u_2, u_1)$,*
   (c) *for any $u_1, u_2, u_3 \in U$:*
   $$DISM_{\mathbf{A}}(u_1, u_3) \leq DISM_{\mathbf{A}}(u_1, u_2) + DISM_{\mathbf{A}}(u_2, u_3).$$
2. *For any $u_1, u_2, u_3, u_4 \in U$, if $DISM_{\mathbf{A}}(u_1, u_2) < DISM_{\mathbf{A}}(u_3, u_4)$ then we say that objects from the pair (u_3, u_4) are more different than objects from the pair (u_1, u_2), relatively to $DISM_{\mathbf{A}}$.*
3. *If any $u_1, u_2 \in U$ satisfies $DISM_{\mathbf{A}}(u_1, u_2) = 0$ then we say that objects from the pair (u_1, u_2) are not different, relatively to $DISM_{\mathbf{A}}$, i.e., they are indiscernible, relatively to $DISM_{\mathbf{A}}$.*
4. *If any $u_1, u_2 \in U$ satisfies $DISM_{\mathbf{A}}(u_1, u_2) = 1$ then we say that objects from the pair (u_1, u_2) are completely different, relatively to $DISM_{\mathbf{A}}$.*

Let us notice that the above dissimilarity function is not a *metric* (distance) but a *pseudometric*. The reason is that the first metric condition is not satisfied which in the case of the $DISM_{\mathbf{A}}$ function would state that the distance between the pair of objects is equal to 0 if and only if they are the same objects. This condition is not satisfied because of the possibility of existence of non-one-element abstraction classes of the relation $IND_{\mathbf{A}}(A)$, that is, because of the possibility of repetition of objects in the set U.

We present an example of dissimilarity function of pairs of objects of information system.

Example 5. Let $\mathbf{A} = (U, A)$ be an information system $\mathbf{A} = (U, A)$, where $A = \{a_1, ..., a_m\}$ is the set of binary attributes. We define the dissimilarity function of pairs of objects in the following way:

$$\forall (u_1, u_2) \in U \times U : DISM_{\mathbf{A}}(u_1, u_2) = \frac{card(\{a \in A : a(u_1) \neq a(u_2)\})}{card(A)}. \qquad \square$$

Let us notice, that the dissimilarity function defined above is based on a widely known and introduced by Hamming measurement of dissimilarity of two sequences of the same length expressing number of places (positions) on which these two sequences differ.

Now, we can define the neighborhood language.

Definition 8 (*A neighborhood language*)**.** *Let $\mathbf{A} = (U, A)$ be an information system. A neighborhood language for the information system $\mathbf{A}$ (denoted by $NL(\mathbf{A})$ or NL-language, when $\mathbf{A}$ is fixed) is defined in the following way:*

- *the set $AL_{NL}(\mathbf{A}) = U \ \cup \ (0, 1] \ \cup \ \{\neg, \vee, \wedge\}$ is an alphabet of the language $NL(\mathbf{A})$,*
- *expressions of the form (u, ε), where $u \in U$ and $\varepsilon \in (0, 1]$ called as neighborhoods of objects, are atomic formulas of the language $NL(\mathbf{A})$.*

Now, we determine the semantics of language $NL(\mathbf{A})$. The language $NL(\mathbf{A})$ formulas may be treated as the descriptions of object occurring in system $\mathbf{A}$.

Definition 9. *Let $\mathbf{A} = (U, A)$ be an information system and $DISM_{\mathbf{A}}$ be a dissimilarity function of pairs of objects from the system $\mathbf{A}$. The satisfiability of an atomic formula $\phi = (u_0, \varepsilon) \in NL(\mathbf{A})$ by an object $u \in U$ from table $\mathbf{A}$ relative to dissimilarity function $DISM_{\mathbf{A}}$ (denoted by $u \models_{NL(\mathbf{A})} \phi$), is defined in the following way:*

$$u \models_{NL(\mathbf{A})} (u_0, \varepsilon) \Leftrightarrow DISM_{\mathbf{A}}(u_0, u) \leq \varepsilon.$$

Each of formula of languages GDL or NL describes a certain set of objects which satisfy this formula (see Fig. 10). According to Definitions 5 and 8 a set of such objects is included in a set of objects U. However, it is worth noticing that these formulas may be satisfied by objects from outside the set U, that is, belonging to an extension of the set U (if we assume that attribute values on such objects can be received) (see Fig. 10).

An explanation is needed if it comes to the issue of defining pairs of objects in an information system with a dissimilarity function. For information systems many such functions may be defined applying various approaches. A review of such approaches may be found in, *e.g.*, [162, 163, 164, 165, 166, 167, 168, 169, 170, 171]). However, the approaches known from literature usually do not take into account the full specification of a specific information system. That is why in a general case the dissimilarity function of a pair of objects should be defined by experts individually for each information system on the basis of domain knowledge. Such a definition may be given in the form of an arithmetical expression (see Example 5). Very often, however, experts in a given domain are not able to present such an expression

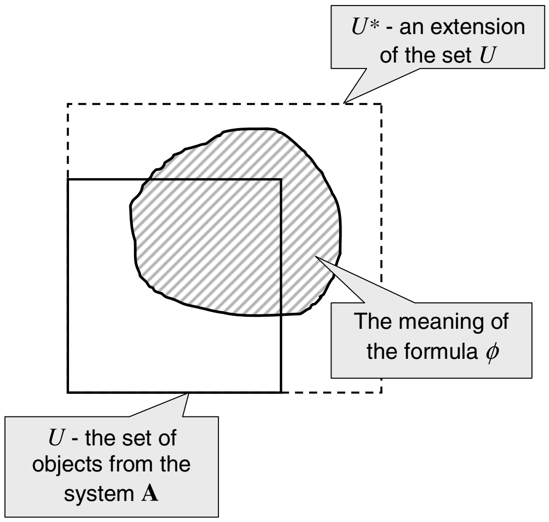

$U*$ - an extension of the set U
The meaning of the formula ϕ
U - the set of objects from the system $\mathbf{A}$

Fig. 10. The illustration of the meaning of a given formula

and content themselves with presenting a set of value examples of this function, that is, a set of pairs of objects labeled with a dissimilarity function value which exists between these objects. In this last case, defining dissimilarity function requires approximation with the help of classifiers. The classifier approximating the dissimilarity function are called dissimilarity classifier of pairs of objects for an information system.

Definition 10. *Let* $\mathbf{A} = (U, A)$ *be an information system* $\mathbf{A} = (U, A)$ *(where* $A = \{a_1, ..., a_m\}$*) and* $DISM_{\mathbf{A}}$ *is a given dissimilarity function of pairs of objects from the system* $\mathbf{A}$*.*

1. *A dissimilarity function table for the system* $\mathbf{A}$ *relatively to the dissimilarity function* $DISM_{\mathbf{A}}$ *is a decision table* $\mathbf{A}_D = (U_D, A_D, d)$*, where:*
   - $U_D \subseteq U \times U$*,*
   - $A_D = \{b_1, ..., b_m, b_{m+1}, ..., b_{2m}\}$*, where attributes from* A_D *are defined in the following way:*

$$\forall u = (u_1, u_2) \in U_D \ \forall b_i \in A_D : \ b_i(u) = \begin{cases} a_i(u_1) & i \leq m \\ a_{i-m}(u_2) & otherwise. \end{cases}$$

   - $\forall u = (u_1, u_2) \in U_D : d(u) = DISM_{\mathbf{A}}(u_1, u_2)$.
2. *If* $\mathbf{A}_D = (U_D, A_D, d)$ *is the dissimilarity function table for the system* $\mathbf{A}$ *then any classifier for the table* $\mathbf{A}_D$ *is called a dissimilarity classifier for the system* $\mathbf{A}$*. Such classifier is denoted by* $\mu_{DISM_{\mathbf{A}}}$*.*

Let us notice that in the dissimilarity table of the information system $\mathbf{A}$ there do not exist all the possible pairs of objects of system $\mathbf{A}$, but only a certain chosen subset of the set of these pairs. This limitation is necessary, for the number of pairs of $U \times U$ product may be so large that the expert is not able to give all the values of decision attribute d for them. That is why in the dissimilarity table there are usually only found pairs chosen by the expert which represent typical cases of dissimilarity function determining which may be generalized with the help of a classifier.

The dissimilarity classifier may serve determining the value of dissimilarity function for the pair of objects from the information system. According to Definition 10 such pairs come from set $U \times U$, that is, they are pairs of objects from a given information system $\mathbf{A}$. However, it should be stressed that the dissimilarity classifier may determine the values of the dissimilarity function for the pairs of objects which do not belong to system $\mathbf{A}$, that is, those which belong to extension of $\mathbf{A}$. Hence, dissimilarity classifiers may be treated as a way to define concepts (new two-argument relations).

The described approach to the measure of dissimilarity is applied in this paper to the measure of dissimilarity between objects in information systems (see Section 6.7 and Section 6.19), between states in planning graphs (see Section 7.9) and plans (see Section 7.20).

4.8 Types of Complex Objects

In a given complex dynamical system there may occur many different complex objects. The collection of all such objects may be represented with the help of information system, where the set of this system's objects correspond with the objects of this collection and the attributes of this system describe the properties of complex objects from the collection and more precisely the properties of relational structures representing individual objects of this collection. Such a system for a given complex dynamical system we call in this paper a *total information system* (TIS) for a given complex dynamical system.

Attributes of the system TIS may be sensor attributes or they are defined in an established language which helps to express the properties of complex objects (see Section 4.7). To the attribute set of the system TIS one may add the binary decision attribute representing the concept describing an additional property of complex objects. The decision attribute may be further approximated with the help of attributes available from the system TIS (see Section 4.7).

However, in practice the concepts which are examined are defined only in the set of complex objects of a certain type occurring in a given complex dynamical system. In the example concerning the traffic (see Example 1) such a concept may concern only cars (*e.g.*, safe overtaking of one car by another), whereas in the example concerning patient treatment (see Example 2), the examined concepts may concern the treatment of infants only, not other people like children, adults or the elderly whose treatment differs from the treatment of infants.

Therefore, we need a mechanism which enable to appropriate selection of complex objects, and more precisely relational structures which they represent and in which we are interested at the moment. In other words, we need a method which enable to select objects of a certain type from the system TIS.

In the paper, we propose a method of adding a binary attribute to TIS to define the types of complex objects, and more precisely the types representing the objects of relational structures. The value YES of such an attribute in a given row, means that the given row represents the complex object that is of the examined type, whereas value NO means that the row represents a complex object which is not of the examined type. The attributes defining types may be defined with the help of attributes from the system TIS in the language GDL or any other language in which the attributes form the system TIS were defined.

The example below shows how the attributes defining the types of complex objects may be defined.

Example 6. Let us assume that in a certain hospital in the children's ward there was applied information system $\mathbf{A} = (U, A)$ to represent the information about patients' treatment, such that $U = \{u_1, ..., u_n\}$ and $A = \{a_1, ..., a_m, a_{age}, a_t, a_{id}\}$. Each object of this system represents medical parameters of a certain child in one day of his/her hospitalization. Attributes $a_1, ..., a_m$ describe medical parameters of the patient (examination results, diagnoses, treatments, medications, etc.), while the attribute a_{age} represents the age of patient (a number of days of life), the a_t attribute represents the value of a time unit (a number of days) which has elapsed since the first observation of the patient and the attribute a_{id} provides identifiers of patients. If system $\mathbf{A}$ is treated as the total information system for a complex dynamical system understood as a set of all patients, then the "infant" type of patient (it is a child not older than 28 days) labeled with T_{inf} may be defined with the help of formula $(a_{age} \leq 28)$. □

A slightly more difficult situation appears in the case of the information system from Example 1, when we want to define the *passenger car* type of object. A written description of the formula defining such a type may be as follows: *the object is perceived as a rectangle whose length is two to five times bigger than its width, and the movement of the object takes place in the direction parallel to the longer side of the rectangle.* It is easy to see that in order to define such a formula the information system from Example 1 would have to be complemented with sensor attributes determining the coordinates of the characteristic points of the object for determining its sizes, shape and movement direction.

If we define an additional attribute by determining the type of object in the system TIS, then we can select information subsystem in which all objects will have the same value of this attribute. Using a subsystem selected in such a way one may analyze concepts concerning the established type of objects. Obviously, during the approximation of these concepts the attribute determining the type according to which an object selection was previously performed is useless, because its value is the same for all selected objects. Therefore, the attributes defining the type of object are not used to approximate concepts, but only to an initial selection of objects for the need of concept approximation.

In a given complex dynamical system there may be observed very different complex objects. The diversity of objects may express itself both through the degree of spatial complexity and by the spatio-temporal complexity (see Section 4.3). Therefore, in a general case it should be assumed that in order to describe the properties of all complex objects occurring in a given dynamical system, many languages must be used. For instance, to describe the properties of a single vehicle at a single time point, the information obtained directly from the sensors are usually used (*e.g.*, speed, location), to describe the properties of a vehicle observed for a certain period of time (time window), a language may be used which enables to define the so-called temporal patterns observed in time windows (see Section 6.6), whereas in order to describe the properties of groups of vehicles a language may be used which enable to define temporal patterns observed in the sequences of time windows (see Section 6.17). Moreover, it usually happens that not each of these languages is appropriate to express the properties of all complex objects occurring in a given complex dynamical system. For example, if we want to apply the language of temporal patterns to determine the properties of a vehicle at a single time point, then it is not feasible because this language requires information about the vehicle collected in the whole time window not at a single time point. Therefore, the approach to recognizing types of complex objects described above must be complemented. Namely, the attributes defining types of complex objects, apart from the values YES and NO mentioned before, may also have the *UNKNOWN* value. This value means that for a given complex object it is not possible to compute correctly the value of an attribute.

Summarizing, if we examine complex objects from a certain complex dynamical system and claim that a given complex object u is a complex object of type T, then it means that in the total information system constructed for this system there exists such attribute a_T that it takes the value YES for object u. One may also say that a given complex object u is not a complex object of type T which means that attribute a_T corresponding with type T takes the value NO for object u. The value of attribute a_T for object u may also take the value *UNKNOWN* which in practice also means that object u is not of type T.

A given complex object may be an object of many types, because there may exist many attributes identifying types in TIS which take the value YES for this object. For example, in the information system from Example 6 the type of object T_r may be defined which can be described in words as *the patient recently admitted to hospital* (that is admitted not earlier than three days ago) with the help of formula ($a_t \leq 3$). Then, the infant admitted to hospital for treatment two days ago is a patient of both type T_{inf} and T_r.

Finally, let us notice that the above approach to determining types of objects may be applied not only to complex objects which were observed at the moment of defining the formula determining the type, but also to those complex objects which appeared later, that is, belong to the extension of the system TIS. It results from the properties of formulas of the language GDL which define the types of objects in the discussed approach.

4.9 Patterns

If an attribute of a complex object collection is a binary attribute (it describes a certain concept), then the formula enables to determine its values is usually called a pattern for the concept. Below, we present a pattern definition assuming that there is given a language $\mathcal{L}$ defining features of complex objects of a determined type, defined using Tarski's approach (see, *e.g.*, [304]).

Definition 11 (*A pattern*)**.** *Let S be a collection of complex objects of a fixed type T. We assume $C \subseteq S$ is a concept and $\mathcal{L}$ is language of formulas defining (under a given interpretation of $\mathcal{L}$ defined by a satisfiability relation) features of complex objects from the collection S (i.e., subsets of S defined by formulas under the given interpretation).*

1. *A formula $\alpha \in \mathcal{L}$ is called a pattern for concept C explained in the language $\mathcal{L}$ if exists $s \in S$ such that $s \in C$ and $s \models_{\mathcal{L}} \alpha$ (s satisfies α in the language $\mathcal{L}$).*
2. *If $s \models_{\mathcal{L}} \alpha$ then we say that s matches pattern α or s supports pattern α. Otherwise s does not match pattern or does not support pattern α.*
3. *A pattern $\alpha \in \mathcal{L}$ is called exact relative to the concept C when for any $s \in S$, if $s \models_{\mathcal{L}} \alpha$ then $s \in C$. Otherwise, a pattern α is called inexact.*
4. *The number:*
$$support(\alpha) = card(|\alpha|_{\mathcal{L}})$$
is called the support of the pattern α.
5. *The confidence of the pattern α relatively to the concept C we denote as $confidence_C(\alpha)$ and define in the following way:*
$$confidence_C(\alpha) = \frac{card(\{s \in C : s \models_{\mathcal{L}} \alpha\})}{support(\alpha)}.$$

Thus patterns are simple but convenient way of defining complex object properties and they may be applied to information system construction representing complex object collections.

Despite the fact that according to Definition 11, patterns are supposed to describe complex object properties belonging to a given complex object collection S, they may also describe complex object properties from outside of the S collection. However, they always have to be complex objects of the same type as the objects gathered in collection S.

Patterns may be defined by experts on the basis of domain knowledge. In such a case the expert must define a needed formula in a chosen language which enables to test objects on their membership to the pattern. In a general case, patterns may be also approximated with the help of classifiers. In this case, it is required from the expert to give only examples of objects belonging to the pattern and counterexamples of objects not belonging to the pattern. Then, however, attributes which may be used to approximate the pattern are needed.

Sometimes in an information system representing a complex object collection one of the attributes is distinguished. For example, it may represent a concept distinguished by the expert which requires approximation using the rest of the attributes. Then such an information system is called a decision table (see Section 2.1).

The decision table constructed for a complex object collection may be useful in classifier construction which ensures the approximation of the distinguished decision attribute. The approximation may be performed with the help of classical classifiers (see Section 2) or stratifying classifiers (see Section 3).

As we wrote before, language formulas serving to define complex object properties may be satisfied by complex objects from outside of a given collection of complex objects. Thus, for any complex object being of the same type as complex objects from a given collection, it may be classified using the above mentioned classifier.

4.10 Approximation of Concepts from Ontology

The method of using ontology for the approximation of concepts presented in this section consists of approximating concepts from the higher level of ontology using concepts from the lower levels.

For the concepts from the lowest hierarchical level of ontology (sensor level), which are not dependent on the rest of the concepts, it is assumed that there are also available the so called *sensor attributes* which enable to approximate these concepts on the basis of applied positive and negative examples of objects.

Below, we present an example of concept approximation using sensor attributes in a certain ontology.

Example 7. Let us consider the ontology from Fig. 11. Each vehicle satisfying the established condition expressed in a natural language belongs to some concepts of this ontology. For example, to the concept of *Safe overtaking* belong vehicles which overtake safely, while to the concept of *Possibility of safe stopping before the crossroads* belong vehicles whose speed is so small that they may safely stop before the crossroads. Concepts of the lowest ontology level that is *Safe distance from the opposite vehicle during overtaking*, *Possibility of going back to the right lane*, *Possibility of safe stopping before the crossroads*, *Safe distance from the front vehicle*, *Forcing the right of way* and *Safe distance from the front vehicle* are sensor concepts, that is, they may be approximated directly using sensor data. For instance, the concept of *Possibility of safe stopping before the crossroads* may be approximated using such sensor attributes as *vehicle speed*, *vehicle acceleration*, *distance to the crossroads*, *visibility* and *road humidity*. □

On the higher levels of ontology, however, sensor attributes may not be used directly to approximate concepts because the semantical distance of approximated concepts from sensor attributes is too large and they are defined on different levels of abstraction. For example, if we wish to approximate the concept of *safe driving* on the higher level and on the sensor level we have at our disposal only attributes giving simple parameters of vehicle driving (that is, location, speed, acceleration, etc.), then it is hard to expect that these parameters allow to make the approximation of such a complex concept as *safe driving* possible. That is why in this paper we propose a method of approximating the concept from the higher level of ontology only with the help of concepts from the ontology level that is lower by one level, which are closer to the concept under approximation

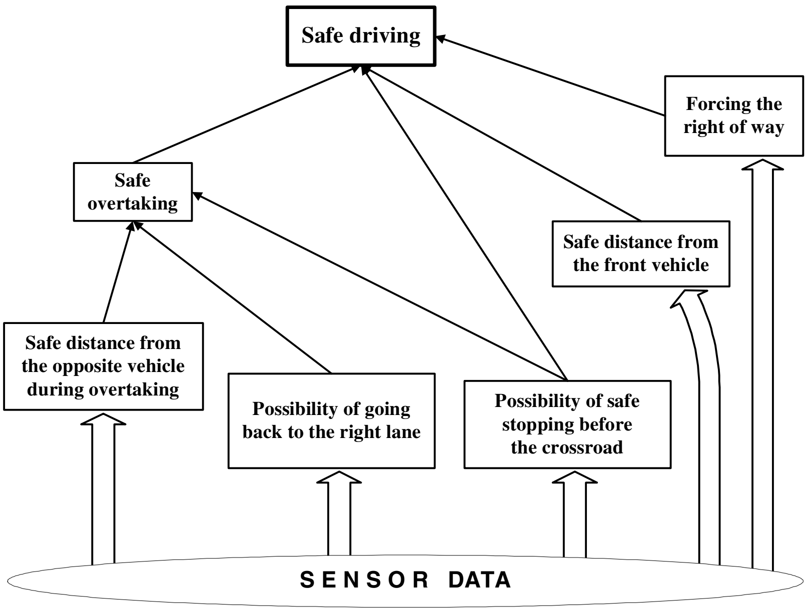

Safe driving
Forcing the right of way
Safe overtaking
Safe distance from the front vehicle
Safe distance from the opposite vehicle during overtaking
Possibility of going back to the right lane
Possibility of safe stopping before the crossroad
SENSOR DATA

Fig. 11. An ontology as a hierarchy of concepts for approximation

than the sensor data. The proposed approach to the approximation of concepts of the lower level is based on an assumption that a concept from the higher ontology level is "not too far" semantically from concepts lying on the lower level of ontology. "Not too far" means that it can be expected that it is possible to approximate a concept from the higher level of ontology using concepts from the lower level for which classifiers have already been built.

The proposed method of approximating concepts of the higher ontology level is based on constructing a decision table for a concept on the higher ontology level whose objects represent positive and negative examples of the concept approximated on this level; and at the same time a stratifying classifier is constructed for this table. In this paper, such a table is called *a concept approximation table* of the higher ontology level concept.

One of the main problems related to construction of the concept approximation table of the higher ontology level concept is providing positive and negative examples of the approximated concept on the basis of data sets. It would seem that objects which are the positive and negative examples of the lower ontology levels concepts may be used at once (without any changes) for concept approximation on the higher ontology level. If it could be possible to perform, any ontology concepts could be approximated using positive and negative examples available from the data sets. However, in a general case, because of semantical differences between concepts and examples on different levels of ontology, objects of the lower level cannot be directly used to approximate concepts of the higher ontology level. For example, if on a higher level of a concept hierarchy, we have a concept concerning a group of vehicles, and on a lower one concepts concerning single vehicles, then usually the properties of single vehicles (defined in order to approximate concepts of lower levels of ontology) are not sufficient to describe properties of a whole group of vehicles. Difficulties with approximation of concepts on the higher ontology level with the help of object properties from the lower ontology level also appear when on the higher ontology level there are concepts concerning another (*e.g.*, longer) period of time than concepts on the lower ontology level. For example, on the higher level we examine a concept concerning a time window (a certain time period), yet on the lower level they are concepts concerning a certain instant, i.e., a time point (see Section 6).

That is why in this paper we propose a method for constructing objects of an approximation table of the concept from the higher ontology level (that is, positive and negative examples of this concept) by arranging sets of objects which are positive and negative examples of the lower ontology level concepts. These sets must be constructed in such a way, that the properties of these sets considered together with relationships between their elements could be used for the approximation of the higher ontology level concept. However, it should be stressed here that the complex objects mentioned above (being positive and negative examples of concepts from the higher and lower ontology levels) are representation of real-life objects only. In other words, we assume that the relational structures are expressing the result of perception of real-life objects (see Section 4.5 and Fig. 12). Therefore, by the features of complex objects represented with

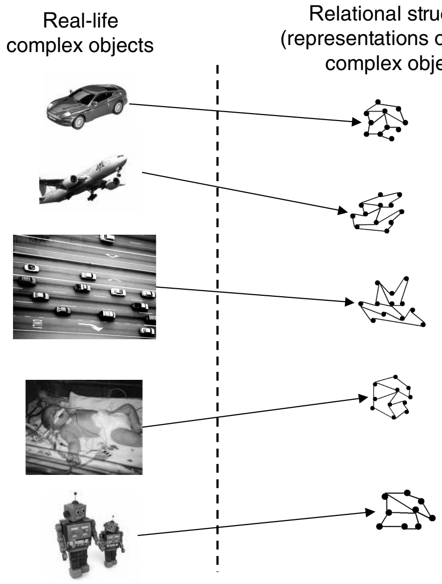

Real-life complex objects
Relational structures (representations of real-life complex objects)

Fig. 12. Real-life complex objects and representations of their structures

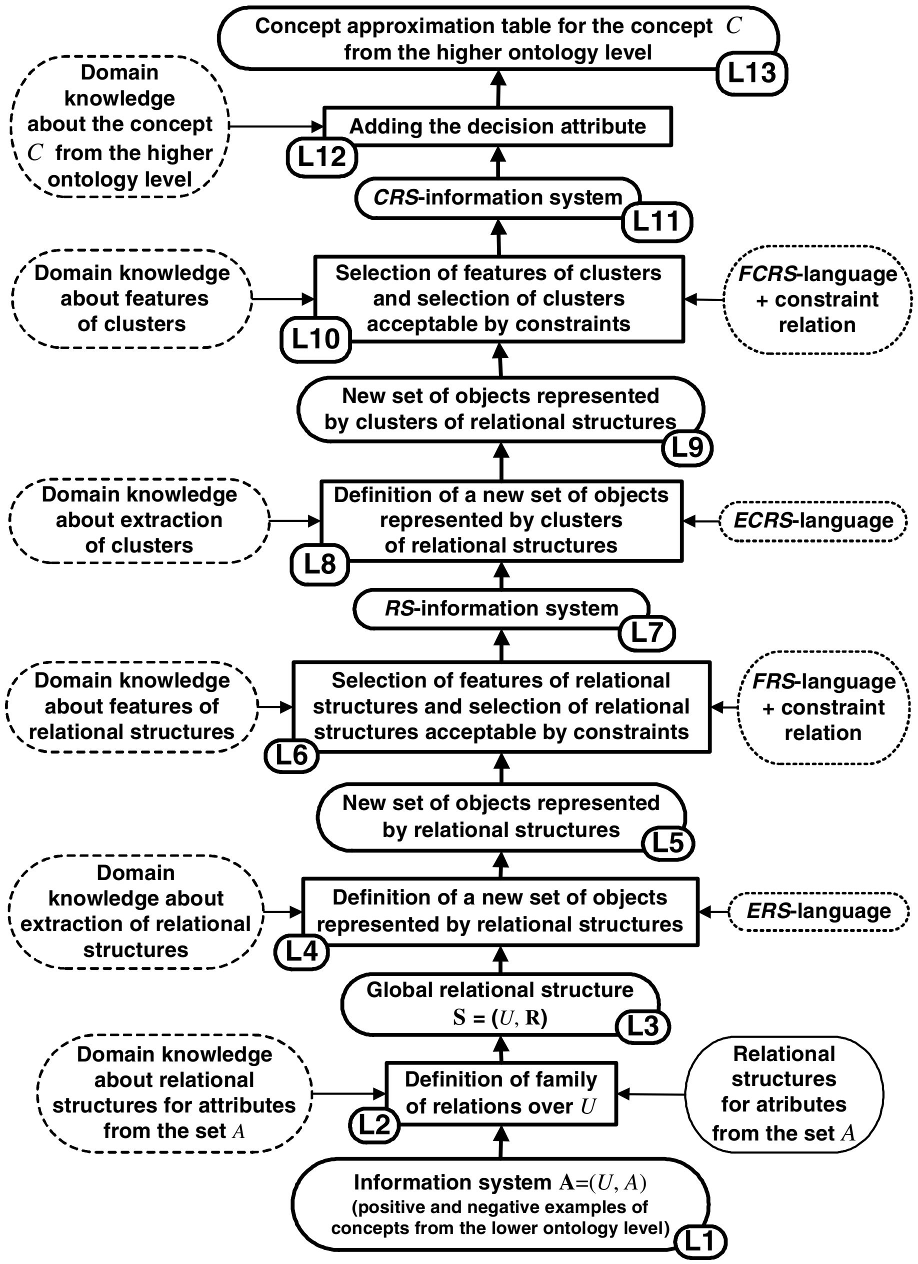

Concept approximation table for the concept C from the higher ontology level
L13
Domain knowledge about the concept C from the higher ontology level
Adding the decision attribute
L12
CRS-information system
L11
Domain knowledge about features of clusters
Selection of features of clusters and selection of clusters acceptable by constraints
L10
FCRS-language + constraint relation
New set of objects represented by clusters of relational structures
L9
Domain knowledge about extraction of clusters
Definition of a new set of objects represented by clusters of relational structures
L8
ECRS-language
RS-information system
L7
Domain knowledge about features of relational structures
Selection of features of relational structures and selection of relational structures acceptable by constraints
L6
FRS-language + constraint relation
New set of objects represented by relational structures
L5
Domain knowledge about extraction of relational structures
Definition of a new set of objects represented by relational structures
L4
ERS-language
Global relational structure S = (U, R)
L3
Domain knowledge about relational structures for attributes from the set A
Definition of family of relations over U
L2
Relational structures for atributes from the set A
Information system A=(U, A) (positive and negative examples of concepts from the lower ontology level)
L1

Fig. 13. The general scheme for construction of the concept approximation table

relational structures we understand the features of these structures. Such features are defined using attributes from information systems from the higher and lower ontology levels.

In Fig. 13, we illustrate the general scheme for construction of the concept approximation table for a given concept C depending in some ontology on concepts from the lower level (relatively to the concept C). In the further part of the subsection, this scheme will be explained in detail.

As we have written before, in this paper we assume that for the concepts of the lower ontology level a collection of objects which are positive and negative examples of these concepts is available. Let us also assume that they are objects of a certain information system $\mathbf{A} = (U, A)$, where attributes from set A represent all available properties of these objects (see label **L1** from the Fig. 13).

It should be stressed here that the information about the membership degree of objects from set U to the concepts from the lower ontology level may serve defining new attributes which are appended to the set A. However, providing such information for a randomly chosen object (also for an object which will appear in the future) requires previous approximation of concepts of the lower level with the help of classical or stratifying classifiers. At this point, we assume that for the concepts of the lower ontology level such classifiers were already constructed, while our aim is to approximate the concept of the higher ontology level. Incidentally, in the simplest case, the concepts of the lower ontology level may be approximated with the help of sensor attributes (see Example 7).

Apart from attributes defined on the basis of the membership of objects to the concepts or to the layers of the concepts, there may be other attributes in set A. For example, it may be an attribute identifying the recording time of values of the remaining attributes from set A for a given object from set U or an attribute unambiguously identifying individual objects or groups of objects from set U.

Objects being positive and negative examples of the lower ontology level concepts can be very often used to define new objects represented by relational structures by using available information about these objects. Relations defined in such structures may be also used to filter (extract) sets of objects or, in a more general case, sets of relational structures or their clusters as new objects for a higher level concept.

Relations among objects may be defined on the basis of attributes from the information system $\mathbf{A}$, with the use of relational structures defined on the value sets of attributes from set A (see label **L2** from the Fig. 13). For example, the value set of attribute V_{a_t} from Example 2 is a subset of the set of integer numbers. Therefore, it is a domain of a relational structure $(V_{a_t}, \{R_{a_t}\})$, where relation R_{a_t} is defined in the following way:

$$\forall (t_1, t_2) \in V_{a_t} \times V_{a_t} : t_1 R_{a_t} t_2 \Leftrightarrow t_1 \leq t_2.$$

Relation R_{a_t} may be in a natural way, generalized to the relation $R_t \subseteq U \times U$ in the following way:

$$\forall (u_1, u_2) \in U \times U : \; u_1 R_t u_2 \Leftrightarrow a_t(u_1) \; R_{a_t} \; a_t(u_2).$$

Let us notice that relation R_t orders in time the objects of the information system from Example 2. Moreover, it is also worthwhile mentioning that for any pair of objects $(u_1, u_2) \in U^\star \times U^\star$ (where $U \subseteq U^\star$) the relation R_t is also defined (if we assume that attribute values on such objects can be received) (see Fig. 10).

Analogously, a relation ordering objects in time on the basis of attribute t from the information system from Example 1 may be obtained.

Obviously, relations defined on the basis of the attributes of information system $\mathbf{A}$ are not always related to the ordering objects in time. The example below illustrates how structural relations may be defined on the basis of the distance between objects.

Example 8. Let us consider an information system $\mathbf{A} = (U, A)$, whose object set $U = \{u_1, ..., u_n\}$ is a finite set of vehicles going from a town T_1 to a town T_2, whereas two attributes d and v belong to the attribute set A. The attribute d represents the distance of a given vehicle from the town T_2 while attribute v represents the speed of a given vehicle. Value sets of these attributes are subsets of the set of real numbers. Besides, the set V_d is a domain of relational structure $(V_d, \{R_d^\varepsilon\})$, where the relation R_d^ε is defined in the following way:

$$\forall (v_1, v_2) \in V_d \times V_d : \; v_1 \; R_d^\varepsilon \; v_2 \Leftrightarrow |v_1 - v_2| \leq \varepsilon,$$

where ε is a fixed real number greater than 0.

Relation R_d^ε may be in a natural way, generalized to the relation $R_\varepsilon \subseteq U \times U$ in the following way:

$$\forall (u_1, u_2) \in U \times U : \; u_1 R_\varepsilon u_2 \Leftrightarrow d(u_1) \; R_d^\varepsilon \; d(u_2).$$

As we see, a pair of vehicles belongs to relation R_ε when objects are distant from each other by no more than ε. Therefore, relation R_ε we call the *nearness relation* of vehicles and parameter ε is called the *nearness parameter* of vehicles. Relation R_ε may be defined for different values ε. That is why in a general case the number of nearness relations is infinite. However, if it is assumed that parameter ε takes the values from a finite set (*e.g.*, $\varepsilon = 1, 2, ..., 100$), then the number of nearness relations is finite. If R_ε is a nearness relation defined in the set $U \times U$ (where $\varepsilon > 0$), then set of vehicles U is a domain of the pure relational structure $\mathbf{S} = (U, \{R_\varepsilon\})$. The exemplary concepts characterizing the properties of individual vehicles may be *high* (*average*, *low*) speed of the vehicle or *high* (*average*, *low*) distance from the town T_2. These concepts are defined by an expert and may be approximated on the basis of sensor attributes d and v. However, more complex concepts may be defined which cannot be approximated with the help of these attributes. The example of such a concept is *vehicle driving in a traffic jam.* The traffic jam is defined by a number of vehicles blocking one another until they can scarcely move (see, *e.g.*, [305]). It is easy to notice that on the basis of observation of the vehicle's membership to the above mentioned sensor concepts (concerning a single vehicle) and even observation of the value of sensor attributes for a given vehicle, it is not possible to recognize whether the vehicle is driving in a traffic jam or not. It is necessary to examine the neighborhood of a given vehicle and more precisely to check whether there are other vehicles right after and before the examined one. Therefore, to approximate the concept *vehicle driving* *in traffic jam* we need a certain type of vehicle grouping which may be performed with the help of the above mentioned relation R_ε (see Example 9). Let us add that in recognition of the vehicle's membership to the concept *vehicle driving in a traffic jam*, it is also important that the speed of the examined vehicle and the speed of the vehicles in its neighborhood are available. However, to simplify the examples, in this subsection we assume that in recognition of the vehicle's membership to the concept *vehicle driving in a traffic jam* it is sufficient to check the appearance of other vehicles in the neighborhood of a given vehicle and considering the speed of these vehicles is not necessary. □

Thus, for a given information system $\mathbf{A} = (U, A)$ representing positive and negative examples of the lower ontology levels concepts there may be defined a pure relational structure $\mathbf{S} = (U, \mathbf{R})$ (see label **L3** from the Fig. 13). Next, using the relations from family $\mathbf{R}$ a special language may be defined in which patterns are defined which describe sets of objects (new concepts) for the needs of approximation of the higher ontology level concepts (see label **L4** from the Fig. 13). The extracted sets of objects of a lower level are also usually nontrivial relational structures, for the relations determined on the whole set of objects of the lower ontology level in a natural way are defined on the extracted sets. Time windows (see Section 6.4) or sequences of time windows (see Section 6.15) may be such kind of relational structures. In modeling, we use pure relational structures (without functions) over set of objects extracted from the initial relational structures whose domains are sets of objects of lower ontology level. The reason is that these structures are defined by extension of relations structures defined on information about objects of lower ontology level and even if in the latter structures are defined functions then after the extension we obtain relations over objects rather than functions.

Example 9. Let us consider an information system $\mathbf{A} = (U, A)$ from Example 8. Let R_ε be the nearness relation defined in the set $U \times U$ for the fixed $\varepsilon > 0$. Then, the vehicle set U is the domain of relational structure $\mathbf{S} = (U, \{R_\varepsilon\})$ and the relation R_ε may be used to extract relational structures from the structure $\mathbf{S}$. In order to do this we define the family of subsets $F(\mathbf{S})$ of the set U in the following way: $F(\mathbf{S}) = \{N_\varepsilon(u_1), ..., N_\varepsilon(u_n)\}$, where:

$$N_\varepsilon(u_i) = \{u \in U : \; u_i R_\varepsilon u\}, \text{ for } i = 1, ..., n.$$

Let us notice that each set from family $F(\mathbf{S})$ is connected with one of the vehicles from set U. Therefore, each of the sets from family $F(\mathbf{S})$ should be interpreted as a set of vehicles which are distant from the established vehicle u no more than by the established nearness parameter ε. In other words each such set is a vehicle set which are in the neighborhood of a given vehicle, with the established radius of the neighborhood area. For instance, if $\varepsilon = 20$ meters then vehicles u_3, u_4, u_5, u_6, and u_7 belong to the neighborhood of vehicle u_5 (see Fig. 14). Finally, let us notice that each set $N \in F(\mathbf{S})$ is a domain of relational structure $(N, \{R_\varepsilon\})$. Thus, we obtain the family of relational structures extracted from structure $\mathbf{S}$. □

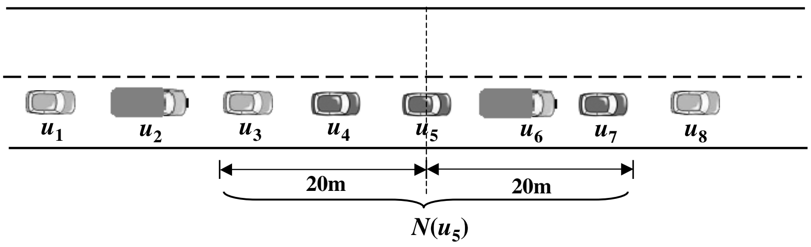


Fig. 14. A vehicle and its neighborhood

The language in which, using the relational structures, we define formulas for expressing extracted relational structures, is called *a language for extracting relational structures* (*ERS*-language). The formulas of *ERS*-language determine type of relational structures, i.e., relational structures which can appear in the constructed information system. These new relational structures represent structure of more compound objects composed out of less compound ones. We call them *extracted relational structures* (see label **L5** from the Fig. 13). In this paper, we use the three following *ERS*-languages:

1. the language assigned to extract trivial relational structures such as presented in Definition 3 and this method of relational structure extraction is used in the case of construction of the concept approximation table using stratifying classifiers (see Section 5.2),
2. the *ETW*-language assigned to extract relational structures which are time windows (see Section 6.4),
3. the *ESTW*-language assigned to extract relational structures which are sequences of time windows (see Section 6.15).

However, the above mentioned process of extracting relational structures is carried out in order to approximate the concept of the higher ontology level with the help of lower ontology level concepts. Therefore, to extract relational structures it is necessary to use information about membership of objects of the lower level to the concepts from this level. Such information may be available for any tested object thanks to the application of previously created classifiers for the lower ontology level concepts (see Section 6.4 and Section 6.15).

For relational structures extracted using *ERS*-language features (properties, attributes) may be defined using a specially constructed language, that we call *a language for definnig features of relational structures* (see label **L6** from the Fig. 13). The *FRS*-language leads to an information system whose objects are extracted relational structures and the attributes are the features of these structures. Such system will be called *an information system of extracted relational structures* (*RS*-information system) (see label **L7** from the Fig. 13). However, from the point of view of domain knowledge, not all objects (relational structures) extracted using *ERS*-language are appropriate to approximation of a given concept of the higher level of ontology. For instance, if we approximate the concept of *safe overtaking*, it is reasonable to use objects representing vehicles examples that are in the process of overtaking maneuver, for using objects representing vehicles which are not in the process of an overtaking maneuver, nothing help to recognize the pairs of vehicles which take part in a safe overtaking with the pairs of vehicles which overtake unsafely.

For the above reason, that is, to eliminate objects which are unreal or are unreasonable, there are defined the so-called *constraints* which are formulas defined on the basis of object features used to create attributes from the RS-system. The constraints determine which objects may be used in order to obtain a concept example from the higher level and which cannot be used (see label **L6** from the Fig. 13). In this paper constraints are represented by a constraint relation and are defined as a formula of the language GDL (see Definition 5) on the basis of attributes appearing in the system RS-system.

The example below illustrates how RS-information systems may be defined.

Example 10. Let us consider an information system $\mathbf{A} = (U, A)$, a relational structure $\mathbf{S} = (U, \{R_\varepsilon\})$ and a family $F(\mathbf{S})$ extracted from relational structure $\mathbf{S}$ (see Example 9). We construct an information system $\overline{\mathbf{F}} = (F(\mathbf{S}), \overline{A})$ such that $\overline{A} = \{\overline{a}_f, \overline{a}_b\}$, where for any $\overline{u} = N_\varepsilon(u) \in F(\mathbf{S})$ a value $\overline{a}_f(\overline{u})$ is the number of vehicles in the neighborhood $N_\varepsilon(u)$ going in the right lane before vehicle u and $\overline{a}_b(\overline{u})$ is the number of vehicles in the neighborhood $N_\varepsilon(u)$ going in the right lane behind vehicle u. Let us notice that attributes of set $\overline{A}$ were chosen in such a way that the objects from information system $\overline{\mathbf{F}}$ are relevant to approximate the concept *vehicle driving in a traffic jam.* For example, if $\varepsilon = 20$ meters and for the object $\overline{u} \in F(\mathbf{S})$ values $\overline{a}_f(\overline{u}) = 2$ and $\overline{a}_b(\overline{u}) = 2$, then vehicle u is driving in a traffic jam (see vehicle u_4 from Fig. 15). Whereas, if $\overline{a}_f(\overline{u}) = 0$ and $\overline{a}_b(\overline{u}) = 0$, then vehicle u is not driving in a traffic jam (see vehicle u_7 from Fig. 15). For the system $\overline{\mathbf{F}}$ we define the following formula:

$$\phi \;=\; ((\overline{a}_f > 0) \;\vee\; (\overline{a}_b > 0)) \in GDL(\overline{\mathbf{F}}).$$

It is easy to notice that formula ϕ is not satisfied only by neighborhoods related to vehicles which definitely not driving in a traffic jam. Therefore, in terms of neighborhood classification to the concept *driving in a traffic jam* these neighborhoods may be called trivial ones. Hence, formula ϕ may be treated as a constraint formula which is used to eliminate the above mentioned trivial neighborhoods from $\overline{\mathbf{F}}$. After such reduction we obtain an RS-information system $\overline{\mathbf{A}} = (\overline{U}, \overline{A})$, where

$$\overline{U} = \{\overline{u} \in F(\mathbf{S}) : \; \overline{u} \models_{GDL(\overline{\mathbf{F}})} \phi\}. \qquad \square$$

Let us notice that the definition of attributes of extracted relational structures leads to granulation of relational structures. For example, we obtain granules of relational structures defined by the indiscernibility relation defined by new attributes.

A question arises, how to construct languages defining features of relational structures, particularly when it comes to approximation of spatio-temporal concepts, that is, those whose recognition requires following the changes of complex objects over time. One of more developed languages of this type is a temporal

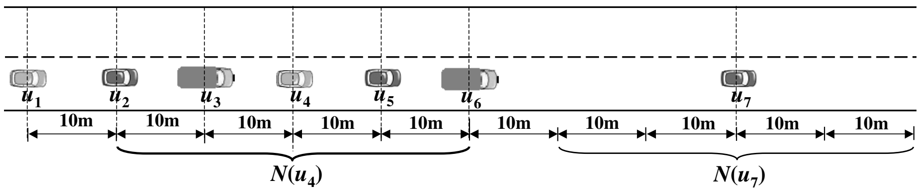


Fig. 15. Two vehicle neighborhoods

logic language. In literature there are many systems of temporal logics defined which offer many useful mechanisms (see, *e.g.*, [183, 184, 185]). Therefore, in this paper, we use temporal logics to define our own languages describing features of relational structures. Especially interesting for us are the elements appearing in definitions of *temporal logics of linear time* (*e.g.*, *Linear Temporal Logic*) and *branching time logic* (*e.g.*, *Branching Temporal Logic*).

Temporal logic of linear time assumes that time has a linear nature, that is, one without branches. In other words, it describes only one world in which each two events are sequentially ordered. In linear time logics there are the following four temporal operators introduced: $\Box$, $\Diamond$, $\bigcirc$ and $\mathcal{U}$. Generally speaking, these operators enable us to determine the satisfiability of temporal formulas in a certain time period. Operator $\Box$ (often also marked as G) determines the satisfiability of a formula at all instants (states) of the time period under observation. Operator $\Diamond$ (often marked as F) determines the satisfiability of a formula at least at one instant (state) of the time period under observation. Operator $\bigcirc$ (often marked as X) determines the satisfiability of a formula at an instant (state) right after the instant of reference. Finally, operator $\mathcal{U}$ (often marked as U) determines the satisfiability of a formula until another formula is satisfied. Therefore, linear time temporal logics may be used to express object properties which aggregate behavior of complex objects observed over a certain period of linear time, *e.g.*, features of time windows or features of temporal paths in behavioral graphs (see Section 6.6 and Section 6.17).

Temporal logic of branching time, however, assumes that time has a branching nature, that is, at a given instant it may branch itself into parallel worlds representing possible various future states. In branching time logics there are two additional path operators A and E introduced. They enable us to determine the satisfiability of temporal formulas for various variants of the future. The first operator means that the temporal formula, before which the operator occurs, is satisfied for all variants of the future. The second, however, means the formula is satisfied for a certain future. Path operators combined with the three G, F and X temporal logics operators give six possible combinations: AG, AF, AX, EG, EF and EX. These combinations give opportunities to describe multi-variant, extended over time behaviors. Therefore, temporal logics of branching time may be used to express such complex object properties that aggregate multi-variant behaviors of objects changing over time (*e.g.*, features of clusters of time windows or features of clusters of temporal paths in behavioral graphs) (see Section 6.8 and Section 6.19).

We assume, that in extracted relational structures the time flow has a linear character. Therefore, languages using elements of temporal logics with linear time are applied to define their features. In this paper, we use the three following languages defining features of extracted relational structure:

1. the language assigned to define features of trivial relational structure such as in Definition 3 - this method of defining features of relational structures is applied together with extraction of trivial relational structure (see Definition 3) and is based on the usage of features of objects taken from information system as features of relational structures after extraction (objects in a given information system and elements of domains of extracted from this system relational structures are the same) (see Section 5.2),
2. the language FTW using elements of temporal logic language and is assigned to define relational structure properties, which are time windows (see Section 6.6),
3. the language FTP also using elements of temporal logic language, assigned to define relational structure properties, which are paths in behavioral graphs (see Section 6.17).

However, objects of RS-information systems are often not suitable to use their properties for approximating concepts of the higher ontology level. It happens this way because the number of these objects is too large and their descriptions are too detailed. Hence, if they are applied to approximate the concept from the higher ontology level, the coverage of the constructed classifier would be too little, that is, the classifier could classify too small number of tested objects. Apart from that, there would appear a problem of computational complexity which means that due to the large number of objects of such information system, the number of objects in the concept approximation table for the structured objects (see further part of this subsection) would be too large in order to construct a classifier effectively.

That is why, a clustering such objects is applied leading to obtaining a family of object clusters (see label **L8** from the Fig. 13).

The example below illustrates in a very simple way how it is possible to define clusters of relational structures.

Example 11. Let $\overline{\mathbf{A}} = (\overline{U}, \overline{A})$ be an RS-information system from Example 10. We are going to define clusters of the vehicles' neighborhoods. For this purpose we propose a relation $\overline{R}_\sigma \subseteq \overline{U} \times \overline{U}$, that is defined in the following way:

$$\forall_{(\overline{u}_1,\overline{u}_2)\in\overline{U}\times\overline{U}}\ \overline{u}_1\overline{R}_\sigma\overline{u}_2 \ \Leftrightarrow\ |\overline{a}_f(\overline{u}_1) - \overline{a}_f(\overline{u}_2)| \leq \sigma \ \wedge\ |\overline{a}_b(\overline{u}_1) - \overline{a}_b(\overline{u}_2)| \leq \sigma,$$

where σ is a fixed integer number greater than 0. As we see, to relation $\overline{R}_\sigma$ belong such pairs of vehicle neighborhoods which differ only slightly (no more than by σ) in terms of attribute values $\overline{a}_f$ and $\overline{a}_b$. Therefore, relation $\overline{R}_\sigma$ is called the *nearness relation of vehicle neighborhoods* and parameter σ is called the *nearness*

parameter of vehicle neighborhoods. The relation $\overline{R}_\sigma$ may be defined for different values σ. That is why in a general case the number of such nearness relations is infinite. However, if it is assumed that parameter σ takes the values from a finite set (*e.g.*, $\sigma = 1, 2, ..., 10$), then the number of nearness relations is finite. Let $\overline{R}_\sigma$ be nearness relation of neighborhoods determined for the established $\sigma > 0$. Then the set of neighborhood of vehicles $\overline{U}$ is the domain of a pure relational structure $\overline{\mathbf{S}} = (\overline{U}, \{\overline{R}_\sigma\})$. The relational structure $\overline{\mathbf{S}}$ is the starting point to extract clusters of vehicle neighborhoods. In order to do this we define the family of subsets $F(\overline{\mathbf{S}})$ of the set $\overline{U}$ in the following way: $F(\overline{\mathbf{S}}) = \{N_\sigma(\overline{u}_1), ..., N_\sigma(\overline{u}_n)\}$,
where:

$$N_\sigma(\overline{u}_i) = \{\overline{u} \in \overline{U} : \ \overline{u}_i \overline{R}_\sigma \overline{u}\}, \text{ for } i = 1, ..., n.$$

Let us notice that each of the set from family $F(\overline{\mathbf{S}})$ is connected with one vehicle neighborhood from the set $\overline{U}$. For any $\overline{u} \in \overline{U}$ the set $N_\sigma(\overline{u})$ will be also denoted by $\overline{\overline{u}}$, for short. Moreover, these sets are interpreted as neighborhood clusters which are distant from the central neighborhood in the cluster no more than the established nearness parameter. In other words, each such family is a vehicles' neighborhood cluster which are close to a given neighborhood, with their established nearness parameter. For instance, if $\varepsilon = 20$ meters and $\sigma = 1$, then neighborhoods $N_\varepsilon(u_3)$, $N_\varepsilon(u_5)$ and obviously neighborhood $N_\varepsilon(u_4)$ belong to the neighborhood cluster $N_\sigma(\overline{u_4})$ (see Fig. 16), whereas the neighborhood $N_\varepsilon(u_7)$ does not belong to this neighborhood cluster. Finally, let us notice that each set $\overline{X} \in F(\overline{\mathbf{S}})$ is a domain of relational structure $(\overline{X}, \{\overline{R}_\sigma\})$. Hence, we obtain the family of relational structures extracted from structure $\overline{\mathbf{S}}$. □

Grouping of objects in system *RS*-system may be performed using chosen by an expert language of extraction of clusters of relational structures, which in this case is called *a language for extracting clusters of relational structures* (*ECRS*-language). The formulas of *ECRS*-language express families of clusters of relational structures from the input *RS*-information systems (see label **L9** from the Fig. 13). Such formulas can be treated as a type of clusters of relational structures which will create objects in a new information system. In *ECRS*-language we may define a family of patterns corresponding to a family of expected clusters. In this paper, the two following *ECRS*-languages are used:

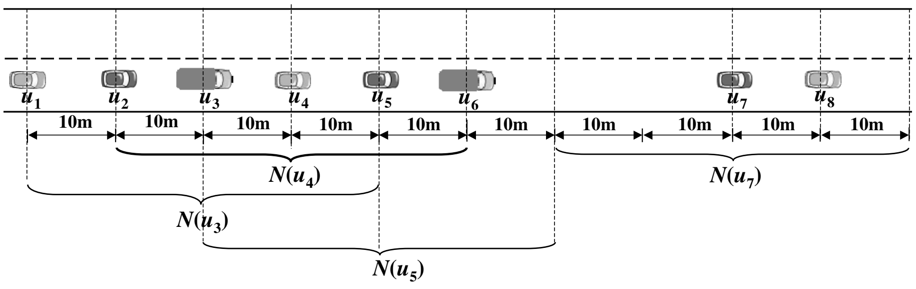


Fig. 16. Four vehicle neighborhoods

1. the language *ECTW* assigned to define relational structure clusters which are time window families (see Section 6.8),
2. the language *ECTP* assigned to define relational structure clusters which are path families in complex object behavioral graphs (see Section 6.19).

For clusters of relational structures extracted in such a way features may be defined using a specially constructed language, that we call *a language for defining features of clusters of relational structures* (*FCRS*-language) (see label **L10** from the Fig. 13). A formula from this language is satisfied (or unsatisfied) on a given clusters of relational structures if and only if it is satisfied for all relational structures from this clusters. The *FCRS*-language leads to an information system whose objects are extracted clusters of relational structures and the attributes are the features of these clusters (see label **L11** from the Fig. 13). Such information system we call *an information system of clusters of relational structures* (*CRS*-information system).

Similarly to the case of the relational structures extracted using *ERS*-language, not all objects (relational structures) extracted using *ECRS*-language are appropriate to approximation of a given concept of the higher level of ontology. Therefore in this case we also define constraints which are formulas defined on the basis of object features used to create attributes from the *CRS*-information system. Such constraints determine which objects may be used in order to obtain a concept example from the higher level and which cannot be used.

The example below illustrates how *CRS*-information systems may be defined.

Example 12. Let $F(\overline{\mathbf{S}})$ be the family extracted from relational structure $\overline{\mathbf{S}}$ (see Example 11). One can construct an information system $\overline{\overline{\mathbf{F}}} = (F(\overline{\mathbf{S}}), \overline{\overline{A}})$, where $\overline{\overline{A}} = \{\overline{\overline{a}}_f, \overline{\overline{a}}_b\}$ and for any $\overline{\overline{u}} \in F(\overline{\mathbf{S}})$ values of attributes $\overline{\overline{a}}_f$ and $\overline{\overline{a}}_b$ are computed as the arithmetical average of values of attributes $\overline{a}_f$ and $\overline{a}_b$ for neighborhoods belonging to the cluster represented by $\overline{\overline{u}}$. The attributes of set $\overline{\overline{A}}$ were chosen in such a way that the objects from set $\overline{\overline{U}}$ are appropriate for approximation of the concept *vehicle driving in a traffic jam.* For example, if $\varepsilon = 20$ meters, $\sigma = 1$ and values $\overline{\overline{a}}_f(\overline{\overline{u}})$ and $\overline{\overline{a}}_b(\overline{\overline{u}})$ are close to 2 then the neighborhoods from cluster represented by object $\overline{\overline{u}}$ contain vehicles which definitely drive in a traffic jam. Whereas, if $\overline{\overline{a}}_f(\overline{\overline{u}})$ and $\overline{\overline{a}}_b(\overline{\overline{u}})$ are close to 0 then the neighborhoods from cluster represented by object $\overline{\overline{u}}$ contain vehicles which definitely do not drive in a traffic jam. For the system $\overline{\overline{\mathbf{F}}}$ we define the following formula:

$$\Phi \;=\; ((\overline{\overline{a}}_f > 0.5) \;\vee\; (\overline{\overline{a}}_b > 0.5)) \in GDL(\overline{\overline{\mathbf{F}}}).$$

It is easy to notice that formula Φ is not satisfied only by such clusters to which belong vehicle neighborhoods definitely not driving in a traffic jam. Therefore, in terms of cluster classification to the concept *driving in a traffic jam* these clusters may be called trivial ones. Hence, formula Φ may be treated as a constraint formula which is used to eliminate the above mentioned trivial clusters from $\overline{\overline{\mathbf{F}}}$.

After such reduction we obtain an CRS-information system $\overline{\overline{\mathbf{A}}} = (\overline{\overline{U}}, \overline{\overline{A}})$, where

$$\overline{\overline{U}} = \{\overline{\overline{u}} \in F(\overline{\mathbf{S}}) : \ \overline{\overline{u}} \models_{GDL(\overline{\overline{\mathbf{F}}})} \Phi \}. \qquad \square$$

Unlike the single relational structures in relational structure clusters the time flow has a branching character because in various elements of a given cluster we observe various variants of dynamically changing reality. Therefore, to define relational structure cluster properties we use elements of temporal logics of branching time language. In this paper, we use the two following languages defining cluster properties:

1. the language $FCTW$ using elements of temporal logics language and assigned to define cluster features which are families of time windows (see Section 6.8),
2. the language $FCTP$ also using elements of temporal logics language assigned to define cluster families which are families of temporal paths in behavioral graphs, that is, sub-graphs of behavioral graphs (see Section 6.19).

Finally, we assume that to each object, acceptable by constraints, an expert adds a decision value determining whether a given object belongs to a higher level approximated concept or not (see label **L12** from the Fig. 13). After adding the decision attribute we obtain the concept approximation table for a concept from the higher ontology level (see label **L13** from the Fig. 13).

The notion of concept approximation table concerning a concept from the higher ontology level for an unstructured complex object may be generalized in the case of concept approximation for structured objects (that is, consisting of parts).

Let us assume that the concept is defined for structured objects of type T which consist of parts being complex objects of types T_1,...,T_k. In Fig. 17 we illustrate the general scheme for construction of the concept approximation table for such structured objects. We see that in order to construct a table for approximating a concept defined for structured objects of type T, CRS-systems are constructed for all types of structured object parts, that is, types T_1,...,T_k (see labels **L3−1**,..., **L3−k** from the Fig. 17). Next, these systems are joined in order to obtain a table of approximating concept of the higher ontology level determined for structured objects. Objects of this table are obtained by arranging (linking) all possible objects of linked information systems (see label **L4** from the Fig. 17). From the mathematical point of view such an arrangement is a Cartesian product of sets of objects of linked information systems. However, from the point of view of domain knowledge not all objects links belonging to such a Cartesian product are possible and reasonable (see [78, 84, 186, 187]). For instance, if we approximate the concept of *overtaking*, it is reasonable to arrange objects of such pairs of vehicles that drive close to each other. For the above reason, there are defined *constraints* which are formulas defined on the basis of properties of arranged objects. The constraints determine which objects may be arranged in order to obtain a concept example from the higher level and which

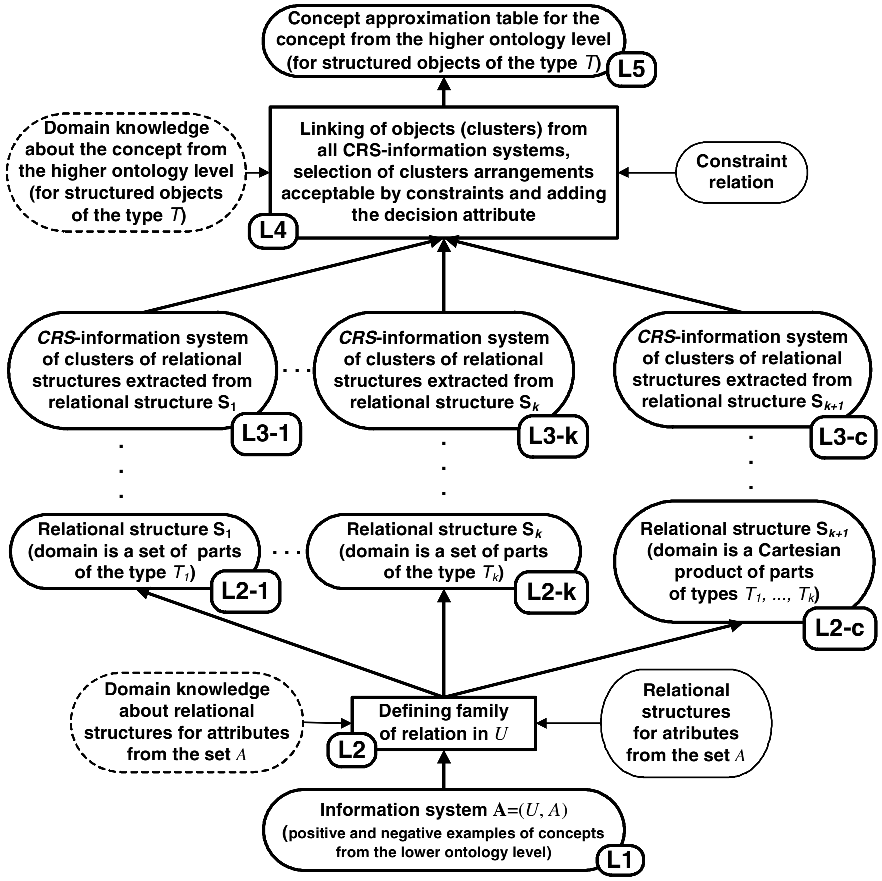

Concept approximation table for the concept from the higher ontology level (for structured objects of the type T)
L5
Domain knowledge about the concept from the higher ontology level (for structured objects of the type T)
Linking of objects (clusters) from all CRS-information systems, selection of clusters arrangements acceptable by constraints and adding the decision attribute
Constraint relation
L4
CRS-information system of clusters of relational structures extracted from relational structure S1
L3-1
CRS-information system of clusters of relational structures extracted from relational structure Sk
L3-k
CRS-information system of clusters of relational structures extracted from relational structure Sk+1
L3-c
Relational structure S1 (domain is a set of parts of the type T1)
L2-1
Relational structure Sk (domain is a set of parts of the type Tk)
L2-k
Relational structure Sk+1 (domain is a Cartesian product of parts of types T1, ..., Tk)
L2-c
Domain knowledge about relational structures for attributes from the set A
Defining family of relation in U
L2
Relational structures for atributes from the set A
Information system A=(U, A) (positive and negative examples of concepts from the lower ontology level)
L1

Fig. 17. The general scheme for construction of the concept approximation table for structured objects

cannot be arranged. Additionally, we assume that to each object arrangement, acceptable by constraints, an expert adds a decision value determining whether a given arrangement belongs to a higher level approximated concept or not (see label **L4** from the Fig. 17).

A table constructed in such a way is to serve a concept approximation determined on a set of structured objects (see label **L5** from the Fig. 17). However, it frequently happens that in order to describe a structured object, apart from describing all parts of this object, a relation between the parts of this object should be described. Therefore, in constructing a table of concept approximation for a structured object, there is constructed an additional CRS-information system whose attributes entirely describe the whole structured object in terms of relations between the parts of this object (see label **L3−c** from the Fig. 17). In approximation of the object concerning structured objects, this system is arranged together with other CRS-information systems constructed for individual parts of the structured objects (see label **L4** from the Fig. 17).

Similarly to the case of the concept approximation table for unstructured objects, the constraint relation is usually defined as a formula in the language GDL (see Definition 5) on the basis of attributes appearing in the obtained table. However, constraint relation may also be approximated using classifiers. In such a case providing examples of objects belonging and not belonging to constraint relation is required (see, *e.g.*, [78]).

The construction of a specific approximation table of a higher ontology level concept requires defining all elements appearing in Figs. 13 and 17. A fundamental problem connected with construction of an approximation table of the higher ontology level concept is, therefore, the choice of four appropriate languages used during its construction. The first language serves the purpose of defining patterns in a set of lower ontology level concept examples which enable the relational structure extraction. The second one enables defining the features of these structures. The third one enables to define relational structure clusters and finally the fourth one the properties of these clusters. All these languages must be defined in such a way as to make the properties of created relational structure clusters useful on a higher ontology level for approximation of the concept occurring there. Moreover, in the case when the approximated concept concerns structured objects each of the parts of this type of objects may require another four of the languages mentioned above.

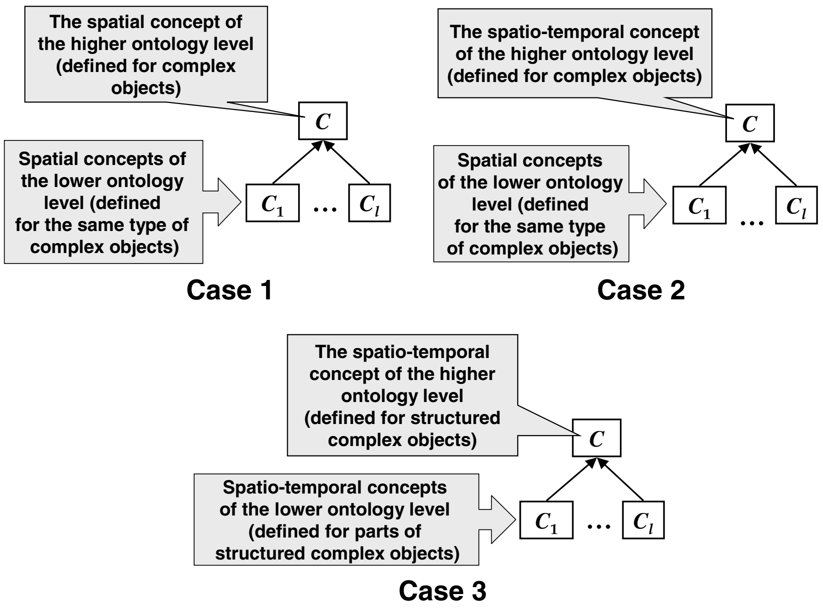

The spatial concept of the higher ontology level (defined for complex objects)
Spatial concepts of the lower ontology level (defined for the same type of complex objects)
C
C1 ... Cl
Case 1
The spatio-temporal concept of the higher ontology level (defined for complex objects)
Spatial concepts of the lower ontology level (defined for the same type of complex objects)
C
C1 ... Cl
Case 2
The spatio-temporal concept of the higher ontology level (defined for structured complex objects)
Spatio-temporal concepts of the lower ontology level (defined for parts of structured complex objects)
C
C1 ... Cl
Case 3

Fig. 18. Three cases of complex concepts approximation in ontology

However, the definition of these languages depends on semantical difference between concepts from both ontology levels. In this paper, we examine the following three situations in which the above four languages are defined in a completely different way (see Fig. 18).

1. The approximated concept C of the higher ontology level is a spatial concept (it does not require observing changes of objects over time) and it is defined on a set of the same objects as lower ontology level concepts (see **Case 1** from Fig. 18). On the lower level we have a concept family: $\{C_1, ..., C_l\}$, that are also spatial concept. Apart from that the concepts $\{C_1, ..., C_l\}$ are defined for unstructured objects without following their changes over time. That is why these concepts are defined on the basis of an object state observation at a single time point or time period established identically for all concepts. For example, the concept C and the concepts C_1,...,C_l may concern the situation of the same vehicle while concept C may be the concept of *Safe overtaking*. On the other hand, to the family of concepts C_1,...,C_l may belong such concepts as: *Safe distance from the opposite vehicle during overtaking*, *Possibility of going back to the right lane* and *Possibility of safe stopping before the crossroads*. The methods of approximation of the concept C for this case are described in Section 5.
2. The concept C under approximation is a spatio-temporal one (it requires observing object changes over time) and it is defined on the set of the same objects as the lower ontology level concepts (see **Case 2** from Fig. 18). On the lower level we have a concept family: $\{C_1, ..., C_l\}$, that are spatial concept. The concept C concerns object property defined in a longer time period than the concepts from the family $\{C_1, ..., C_l\}$. This case concerns a situation when following an unstructured object in order to capture its behavior described by the concept C, we have to observe it longer than it is required to capture behaviors described by concepts from the family $\{C_1, ..., C_l\}$. For example, concepts C_1,...,C_l may concern simple behaviors of a vehicle such as *acceleration*, *deceleration*, *moving towards the left lane*, while the concept C may be a more complex concept: *accelerating in the right lane*. Let us notice that determining whether a vehicle accelerates in the right lane requires its observation for some time which is called a time window. However, determining whether a vehicle increased its speed requires only the vehicle's speed registration at two neighboring instants. Such a case of the concept C approximation is described in Section 6.
3. The approximated concept C is a spatio-temporal one (it requires observing object changes over time) and it is defined on a set of structured objects, while concepts from the family $\{C_1, ..., C_l\}$ are determined on the set of parts of these objects; and at the same time the concept C concerns the structured object's behavior over a longer period of time than concepts from the family $\{C_1, ..., C_l\}$ (see **Case 3** from Fig. 18). This case concerns a situation when following a structured object in order to capture its behavior described by the concept C, we have to observe this object longer than it is required to capture behaviors of single part of this object described by concepts from

the family $\{C_1, ..., C_l\}$. For example, concepts from the family $\{C_1, ..., C_l\}$ may concern complex behaviors of a single vehicle such as *acceleration in the right lane*, *acceleration and changing lanes from right to left*, *decelerating in the left lane*. However, the concept C may be even more complex concept describing a behavior of a group of two vehicles (overtaking and overtaken) over a certain period of time, for example *the overtaking vehicle changes lanes from the right to left one while the overtaken vehicle drives in the right lane*. Let us notice that the behavior described by the concept C is an essential fragment of overtaking maneuver and determining if the group of two vehicles under observation behaved exactly that way requires observation for a certain time of behavior sequence of vehicles taking part in maneuvers such as *accelerating in the right lane*, *accelerating and changing lanes from right to left*, *maintaining a stable speed in the right lane*. This most complex case of the approximation of the concept C also is described in Section 6.

5 Approximating Spatial Concepts from Ontology

In the present subsection, we describe the case of approximating the concept C from the higher ontology level using concepts C_1,...,C_k from the lower ontology level when approximated concept C is defined on the set of the same objects as concepts C_1,...,C_k. Moreover, both concept C and concepts C_1,...,C_k concern object properties without observing their changes over time. In this paper such concepts are called spatial concepts. The example below describes a classic situation of this type resulting from an ontology obtained from a road traffic simulator (see Appendix A).

Example 13. Let us consider a situation when all ontology concepts concern the same type of objects, that is, vehicles. We deal with this type of situation in the case of ontology from Fig. 7. To each concept of this ontology there belong vehicles satisfying a specific condition expressed in a natural language. For example, to the concept of *Safe overtaking* there belong all vehicles which overtake safely, whereas to the concept of *Possibility of safe stopping before the crossroads* these vehicles whose speed is low enough to safely stop before the crossroads. The concepts of the lowest ontology level, that is, *Safe distance from the opposite vehicle during overtaking*, *Possibility of driving back to the right lane*, *Possibility of safe stopping before the crossroads*, *Safe distance from the front vehicle*, *Forcing the right of way* and *Safe distance from the front vehicle* are sensor concepts, that is, they may be approximated directly using sensor data. For example, the concept of *Possibility of safe stopping before the crossroads* may be approximated using such sensor attributes as *the speed of the vehicle*, *the acceleration of the vehicle*, *the distance from the crossroads*, *visibility* and *humidity*. However, concepts of the higher ontology level, that is, *Safe overtaking* and *Safe driving* should be approximated using concepts from the lower ontology level. For example, the concept of *Safe overtaking* may be approximated using the three following concepts: *Safe distance from the opposite vehicle during overtaking*, *Possibility of going back to the right lane* and *Possibility of safe stopping before the crossroads*. □

In order to approximate the higher ontology level concept (for example, the concept of *Safe overtaking* in the above example), an approximation table should be constructed for this concept according to Fig. 13. In order to do this a special language *PEC* is necessary, whose definition we provide in the next subsection.

5.1 Language of Patterns Extracted from a Classifier

If the approximation of the lower level ontology concepts is performed, then for each of these concepts we have at our disposal a classifier which is an algorithm returning for any tested object (that is, relational structure of the lower ontology level) the information about whether this object belongs to the concept or not. This type of information coming from all classifiers approximating lower ontology level concepts may serve the construction of binary attributes which describe crucial properties of the object of the higher ontology level. However, it should not be expected that in a general case the membership of objects to the concept of lower ontology level determines the membership of objects to the concept of higher ontology level. For example, if we assume that the concept of *safe overtaking* depends on the three following concepts: *safe distance from the opposite vehicle during overtaking*, *possibility of driving back to the right lane* and *possibility of safe stopping before the crossroads* (see Fig. 7), then it is hard to expect that a given vehicle overtakes safely only when it belongs to these three concepts. On the other hand, it is hard to expect that if the vehicle does not belong to one of these three concepts, then it definitely does not overtake safely. For example, if the distance from the oncoming vehicle is not safe, that is, a head-on collision of the overtaking and oncoming vehicles is possible, then it cannot be determined that the overtaking is safe. However, there are probably situations when the precise membership of the vehicle to the three concepts above cannot be acknowledged, but the expert will still claim in the natural language that *the overtaking is safe*, or that *the overtaking is almost safe* or *that the overtaking is safe to some degree.* Therefore, in this paper to construct attributes describing object properties from the lower ontology level, we propose stratifying classifiers which must certainly be constructed previously for lower ontology level concepts. This type of attributes inform in a more detailed way about the membership of objects to the lower ontology level concepts and because of that they are more useful to approximate higher level concepts.

Let us, now, define a *language of patterns extracted from a classifier* (*PEC*) which are used to describe object properties which are positive and negative examples of ontology concepts.

Definition 12 (*A language of patterns extracted from a classifier*)**.** *Let us assume that:*

- $\mathbf{A} = (U, A, d)$ *is a decision table, whose objects are relational structures and examples (positive and negative) for some concept C, described by a binary attribute d,*
- μ_C^E *is a stratifying classifier for the concept C, which classifies objects from U to l-layers, denoted be labels from the set $E = \{e_1, ..., e_l\}$, where the following three conditions are satisfied (see also Section 3.1):*

1. *layer e_1 includes objects which, according to an expert, certainly do not belong to concept C (so they belong to a lower approximation of its complement),*
2. *for every two layers e_i, e_j (where $i < j$), layer e_i includes objects which, according to an expert, belong to concept C with a degree of certainty lower the degree of certainly of membership of objects of e_j in U,*
3. *layer e_l includes objects which, according to an expert, certainly belong to concept C, viz., to its lower approximation.*

1. *The language of patterns extracted from a classifier μ_C^E (denoted by $PEC(\mu_C^E)$ or PEC-language, when μ_C^E is fixed) is defined in the following way:*
   - *the set $AL_{PEC}(\mu_C^E) = \{\mu, \in, \neg, \wedge, \vee\} \cup (2^E \setminus \emptyset)$ is an alphabet of the language $PEC(\mu_C^E)$,*
   - *expressions of the form $(\mu \in B)$, for any $B \subseteq E$, are atomic formulas of the language $PEC(\mu_C^E)$.*
2. *The semantics of atomic formulas from the language $PEC(\mu_C^E)$ is defined for any $B \subseteq E$ in the following way:*

$$|\mu \in B|_{PEC(\mu_C^E)} = \left\{u \in U : \mu_C^E(u) \in B\right\}.$$

The issue of defining atomic formulas themselves (expressions of type $\mu \in B$, where $B \subseteq E$) belonging to the sets of formulas of the language mentioned above also requires an explanation. Because concept layers from the set E are ordered, then the formulas of the form $\mu \in B$ are defined with the help of relation $=$, $\neq$, $<$, $\leq$, $>$ and $\geq$. For example, the $\mu = e_2$ formula describes these objects from the set U which the stratifying classifier μ_C^E classifies to the layer marked by e_2, however, the formula of the form $\mu \geq e_3$ describes these objects from the set U which the stratifying classifier μ_C^E classifies to the e_3 layer or higher, that is, to one of the $e_3, e_4, ..., e_l$ layers.

If it is known which stratifying classifier is used to define the language PEC for a specific concept C and if the set of layers E of the approximated concept is known, then we often use simplification of pattern description consisting in replacing (in formulas of the language PEC) the μ symbol with the name of the approximated concept, which enable to simplify the records in pattern presentation for several concepts at the same time. For example, in approximation of concept C using the stratifying classifier μ_C^E (where $E = \{e_1, ..., e_l\}$), pattern $(\mu \geq e_3)$ are recorded as $C \geq e_3$.

Because the language PEC is the language for construction of the structural relation properties, then each formula of that language in a given information system can be called a pattern. Some of these patterns are of great significance in practical applications. Therefore, we give them special names. The first pattern of this type is the so-called *concept layer pattern* which describes objects belonging to one of the concept layers.

Definition 13. *Let C be a concept and μ_C^E be a stratifying classifier for the concept C, which classifies objects to l-layers, denoted be labels from the set $E = \{e_1, ..., e_l\}$ (see conditions from Definition 12). Any pattern of the form $(\mu = e_i)$, where $i \in \{1, ..., l\}$, is called a layer pattern of concept C.*

Each layer pattern may be treated as a simple classifier which can classify objects matching this pattern. Thus, it is possible to select objects which belong to one of the concept layers. However, frequently in practice such accuracy of indicating one layer for a tested object is often not necessary. For example, we may be interested in patterns which describe such layers which certainly do not precede the established layer. These patterns correspond to the situation when we wish to recognize such concepts that belong to the concept with certainty at least equal to the previously established certainty level. For example, if we consider the concept of *safe overtaking* which has six linearly organized layers *"certainly NO"*, *"rather NO"*, *"possibly NO"*, *"possibly YES"*, *"rather YES"* and *"certainly YES"*, then the $\mu \geq$ "*possibly YES*" pattern describes such vehicles that perhaps overtake safely, rather overtake safely and certainly overtake safely. Hence, this pattern may be useful as a classifier which is not too certain. It is easy to change it to $\mu \geq$ "*rather YES*" by this increasing the certainty of its classification.

Due to practical applications of the above patterns we use a special term to call them, which is given in the definition below.

Definition 14. *Let C be a concept and μ_C^E be a stratifying classifier for the concept C, which classifies objects to l-layers, denoted be labels from the set $E = \{e_1, ..., e_l\}$ (see Definition 12). Any pattern of the form $(\mu \geq e_i)$, where $i \in \{1, ..., l\}$, is called a production pattern of concept C.*

The term from the above definition results from the fact that these types of patterns find application in production rule construction (see Section 5.3).

5.2 Concept Approximation Table Using Stratifying Classifiers

Currently, we present a definition of the approximation table of the higher ontology level concept with the help of stratifying classifiers.

Definition 15 (*A concept approximation table using stratifying classifiers*)**.** *Let us assume that:*

- $\mathbf{A} = (U, A)$ *is a given information system of unstructured objects of the fixed type,*
- *C is a concept, dependent in some ontology on concepts C_1,...,C_k, where $C \subseteq U$ and $C_i \subseteq U$, for $i = 1, ..., k$,*
- $\mathbf{T}_i = (U, A_i, d_{C_i})$ *is a decision table constructed for approximation of the concept C_i such that $A_i \subseteq A$ for $i = 1..., k$ and d_{C_i} is a decision attribute which values describe the membership of objects from U to the concept C_i,*
- $\mu_{C_i}^{E_i}$ *is a stratifying classifier for the concept C_i constructed on the basis the table $\mathbf{T}_i$, which classifies objects from the set U to layers, denoted be labels from the set E_i, for $i = 1, ..., k$,*
- $\Phi_i = \{\phi_i^1, ..., \phi_i^{l_i}\}$ *is a family of patterns defined by formulas from the language $PEC(\mu_{C_i}^{E_i})$, which can be used to define new attributes (features) for objects from the set U, for $i = 1..., k$,*

– *$\mathcal{P}_{PEC} = (U, \Phi, \models_{PEC})$ is a property system, where $\Phi = \bigcup_{i=1}^{k} \Phi_i$ and the satisfiability relation $\models_{PEC}$ is defined in the following way:*

$$\forall (u, \phi) \in U \times \Phi : \ u \models_{PEC} \phi \ \Leftrightarrow$$

$$u \models_{PEC(\mu_{C_i}^{E_i})} \phi, \quad \textit{for } i \in \{1, .., k\} \textit{ such that } \phi \in \Phi_i,$$

– *$\mathbf{A}_\Phi = (U, A_\Phi)$ is an information system defined be the property system $\mathcal{P}_{PEC}$,*
– *$\mathbf{R}_C \subseteq U$ is a relation of constraints defined by a formula $\Psi \in GDL(\mathbf{A}_\Phi)$ that is $\forall_{u \in U}\ u \in \mathbf{R}_C \Leftrightarrow u \models_{GDL(\mathbf{A}_\Phi)} \Psi$.*

A concept approximation table using stratifying classifiers for the concept C relatively to concepts C_1,...,C_k is a decision table $\mathbf{A}_C = (U_C, A_C, d_C)$, where:

– *$U_C = \mathbf{R}_C$,*
– *$A_C = A_\Phi$,*
– *the attribute d_C describes membership of objects from the set U to the concept C.*

According to the above definition, the conditional attributes of concept approximation table are constructed on the basis of stratifying classifiers $\mu_{C_1}^{E_1}, ..., \mu_{C_k}^{E_k}$ which were generated for concepts of the lower ontology level C_1,...,C_k and for layer sets E_1,...,E_k. It ought to be stressed, however, that the number of layers and the layout of layers in sets E_1,...,E_k should be chosen in such a way as to serve effective approximation of complex concept C. In order to do this the layers are chosen by an expert on the basis of the domain knowledge or are obtained on the basis of suitably designed layering heuristics (see Section 3).

It is easy to notice that the table of concept approximation from ontology using stratifying classifiers defined above is a special case of a concept approximation table mentioned in Section 4.10.

Let us now go back to Example 7 which concerned approximation of the concept of *Safe overtaking.*

Example 14 (The continuation of Example 13). For concept *Safe overtaking* approximation we wish to construct an approximation table according to Definition 15. First, stratifying classifiers for concepts *Safe distance from the opposite vehicle during overtaking* (C_{SDOV}), *Possibility of going back to the right lane* (C_{PGBR}) and *Possibility of safe stopping before the crossroads* (C_{SSBC}) should be constructed. Next, conditional attributes are constructed which are defined as patterns in the language PEC, individually for each of the three concepts. The choice of appropriate patterns takes place on the basis of domain knowledge. In the simplest case they may be layer patterns for all layers of concepts C_{SDOV}, C_{PGBR} and C_{SSBC}. Next, on the basis of domain knowledge a relation of constraints is established and used to arrange an approximation table for the *Safe overtaking* concept, leaving only objects which belong to this relation. Finally, also on the basis of domain knowledge values of decision attribute from the *Safe overtaking* concept approximation table is established. $\square$

The concept approximation table using stratifying classifiers may be used for building a classifier which ensures approximation of this concept. The approximation may take place using classical classifiers (see Section 2) or stratifying classifiers (see Section 3).

Slightly similar approaches have been successfully applied to approximate concepts in different ontologies (see, *e.g.*, [80, 81, 179, 306, 307]). They have also been applied in ontology from Fig. 7 (see [179, 306]) obtained from the road simulator (see Appendix A). However, in this paper we are more interested in other methods of classifier construction using language PEC which use *production rules*, *productions* and *approximate reasoning schemes*. These methods are described in the next few subsections.

5.3 Production Rules

The production rule (see, *e.g.*, [77, 89, 172, 188, 189, 190, 191, 192, 193]) is a kind of decision rule which is constructed on two adjacent levels of ontology. In the predecessor of this rule there are patterns for the concepts from the lower level of ontology while in the successor the pattern for one concept from the higher level of ontology (connected by relationships with concepts from the rule predecessor).

Definition 16 (*A production rule*). *Let us assume that:*

- $\mathbf{A} = (U, A)$ *is a given information system of unstructured objects of the fixed type,*
- C *is a concept, dependent in some ontology on concepts* C_1,...,C_k, *where* $C \subseteq U$ *and* $C_i \subseteq U$, *for* $i = 1, ..., k$,
- $\mathbf{T}_i = (U, A_i, d_{C_i})$ *is a decision table constructed for approximation of the concept* C_i *such that* $A_i \subseteq A$ *for* $i = 1..., k$ *and* d_{C_i} *is a decision attribute which values describe the membership of objects from* U *to the concept* C_i,
- $\mu_{C_i}^{E_i}$ *is a stratifying classifier for the concept* C_i *constructed on the basis the table* $\mathbf{T}_i$, *which classifies objects from the set* U *to layers, denoted be labels from the set* E_i, *for* $i = 1, ..., k$,
- $\mathbf{A}_C = (U_C, A_C)$ *is a concept approximation table for the concept* C *using stratifying classifiers* $\mu_{C_i}^{E_i}$, *for* $i = 1, ..., k$,
- μ_C^E *is a stratifying classifier for the concept* C, *which classifies objects from the set* U_C *to layers, denoted be labels from the set* E.

1. *If* $p_i \in PEC(\mu_{C_i}^{E_i})$ *is a production pattern for the concept* C_i *(for* $i = 1, ..., k$*) and* $p \in PEC(\mu_C^E)$ *is a production pattern for the concept* C *then any formula of the form:*
$$p_1 \wedge ... \wedge p_k \Rightarrow p \qquad (6)$$
is called a production rule for the concept C *relatively to concepts* $C_1, ..., C_k$ *if and only if the following conditions are satisfied:*
(a) exists at least one object $u \in U_C$ *such that:*
$$u \models_{PEC(\mu_{C_i}^{E_i})} p_i \text{ for } i = 1, ..., k,$$

(b) for any object $u \in U_C$:

$$u \models_{PEC(\mu_{C_1}^{E_1})} p_1 \wedge ... \wedge u \models_{PEC(\mu_{C_k}^{E_k})} p_k \Rightarrow u \models_{PEC(\mu_C^E)} p$$

2. *The first part of production rule (i.e., $p_1 \wedge ... \wedge p_k$) is called a predecessor of production rule, whilst the second part of production rule (i.e., p) is called a successor of production rule.*
3. *The concept from the upper level of production rule (from successor of rule) is called a target concept of production rule, whilst the concepts from the lower level of production rule (from predecessor of rule) are called source concepts of production rule.*

Below, we present an example of production rule.

Example 15. We consider the concept C which depends on concepts C_1 and C_2 (in some ontology). Besides, concepts C, C_1 and C_2 have six linearly organized layers *"certainly NO"*, *"rather NO"*, *"possibly NO"*, *"possibly YES"*, *"rather YES"* and *"certainly YES"*. In Fig. 19 we present an example of production rule for concepts C_1, C_2 and C. This production rule has the following

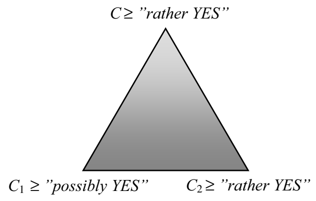


Fig. 19. The example of production rule

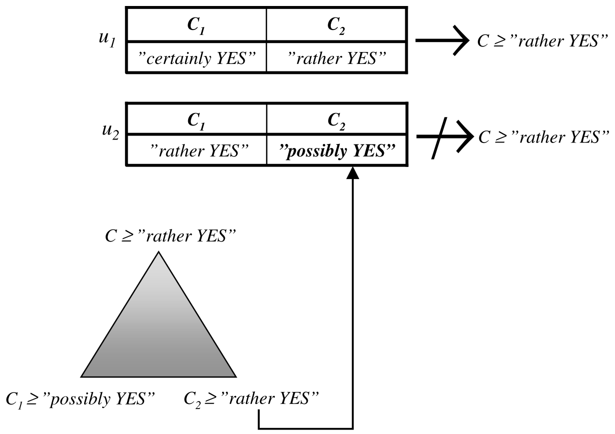


Fig. 20. Classifying tested objects by single production rule

interpretation: if inclusion degree to a concept C_1 is at least *"possibly YES"* and to concept C_2 at least *"rather YES"* then the inclusion degree to a concept C is at least *"rather YES"*. □

A rule constructed in such a way may serve as a simple classifier enabling the classification of objects matching the patterns from the rule predecessor into the pattern from the rule successor. The tested object may be classified by a production rule if it matches all patterns from the production rule predecessor. Then the production rule classifies a tested object to the target (conclusion) pattern.

For example, the object u_1 from Fig. 20 is classified by production rule from Fig. 19 because it matches both patterns from the left hand side of the production rule whereas, the object u_2 from Fig. 20 is not classified by production rule because it does not match the second source pattern of production rule (the value of attribute C_2 is less than *"rather YES"*).

The domain of a given production rule is a set of all objects matching all patterns from the predecessor of this rule.

5.4 Algorithm for Production Rules Inducing

Production rules can be extracted from data using domain knowledge. In this section we present an exemplary algorithm for the production rule inducing. The basic structure of this algorithm's data is a special table called *a layer table* which we define for the approximated concept in ontology.

Definition 17 (*A layer table*). *Let us assume that:*

- $\mathbf{A} = (U, A)$ *is a given information system of unstructured objects of the fixed type,*
- C *is a concept, dependent in some ontology on concepts* C_1,...,C_k, *where* $C \subseteq U$ *and* $C_i \subseteq U$, *for* $i = 1, ..., k$,
- $\mathbf{T}_i = (U, A_i, d_{C_i})$ *is a decision table constructed for approximation of the concept* C_i *such that* $A_i \subseteq A$ *for* $i = 1..., k$ *and* d_{C_i} *is a decision attribute which values describe the membership of objects from* U *to the concept* C_i,
- $\mu_{C_i}^{E_i}$ *is a stratifying classifier for the concept* C_i *constructed on the basis the table* $\mathbf{T}_i$, *which classifies objects from the set* U *to layers, denoted be labels from the set* E_i, *for* $i = 1, ..., k$,
- $\mathbf{A}_C = (U_C, A_C)$ *is a concept approximation table for the concept* C *using stratifying classifiers* $\mu_{C_i}^{E_i}$, *for* $i = 1, ..., k$,
- μ_C^E *is a stratifying classifier for the concept* C, *which classifies objects from the set* U_C *to layers, denoted be labels from the set* E.

A layer table for the concept C *relatively to concepts* C_1,...,C_k *is a decision table* $\mathbf{LT}_C = (U, A, d)$, *where:*

- $U = U_C$
- $A = \{a_{C_1}, ..., a_{C_k}\} \cup \{a_C\}$, *where for any object* $u \in U$ *attributes from the set* A *are defined in the following way:*

- $a_{C_i}(u) = \mu_{C_i}^{E_i}(u)$ *for* $i = 1, ..., k$,
- $a_C(u) = \mu_C^E(u)$.

The layer table for a given concept C, which depends on concepts C_1, ..., C_k in ontology, stores layer labels of objects belonging to the $\mathbf{A}_C$ table.

Example 16. Let us assume that in a certain ontology the concept C depends on concepts C_1 and C_2. Moreover, each of these six concepts has six linearly organized layers: *"certainly NO"*, *"rather NO"*, *"possibly NO"*, *"possibly YES"*, *"rather YES"* and *"certainly YES"*. The Fig. 21 presents a sample table of layers for these concepts. □

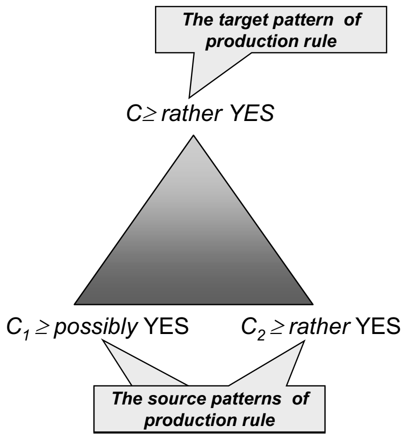
Now, an algorithm of production rule searching may be presented (see Algorithm 5.1). It works on the basis of a layer table and as a parameter requires providing a layer which occurs in the successor of the production rule.

In Fig. 21 we illustrate the process of extracting production rule for concept C and for the approximation layer "*rather YES*" of concept C. Is is easy to see that if from the table $\mathbf{LT}_C$ we select all objects satisfying a_C = "*rather YES*", then for selected objects minimal value of the attribute a_{C_1} is equal to "*possibly YES*" and minimal value of the attribute a_{C_2} is equal to "*rather YES*". Hence, we obtain the production rule:

$$(C_1 \geq \text{"}\textit{possibly YES}\text{"}) \wedge (C_2 \geq \text{"}\textit{rather YES}\text{"}) \Rightarrow (C \geq \text{"}\textit{rather YES}\text{"}).$$

The method of extracting production rule presented above can be applied for various values of attribute a_C. In this way, we obtain a collection of production rules, that we mean as a production (see Section 5.6).

a_{C_1}	a_{C_2}	a_C
certainly YES	*certainly YES*	***certainly YES***
certainly NO	*certainly NO*	*certainly NO*
rather YES	*certainly YES*	***rather YES***
possibly YES	*possibly NO*	*possibly YES*
possibly YES	*possibly NO*	*rather NO*
possibly YES	***rather YES***	***rather YES***
possibly YES	*certainly NO*	*possibly NO*
certainly YES	***rather YES***	***certainly YES***
certainly NO	*possibly NO*	*certainly NO*


The target pattern of production rule
C≥ rather YES
C1 ≥ possibly YES
C2 ≥ rather YES
The source patterns of production rule

certainly NO < rather NO < possibly NO < possibly YES < rather YES < certainly YES

Fig. 21. The illustration of production rule extracting

Algorithm 5.1. Extracting of production rule

```
Input:
  1. concept $C$, dependent on concepts $C_1$,...,$C_k$ (in some ontology).
  2. layer table $\mathbf{LT}_C$ for concept $C$,
  3. label $e$ of layer, that can be placed in the successor of computed
     production rule.
Output: The production rules with the pattern $C \geq e$ placed in its
        successor.
1 begin
2     Select all rows from the table $\mathbf{LT}_C$ in which values of column $a_C$ is
      not less than $e$.
3     Find minimal values $e_1, ..., e_k$ of attributes $a_{C_1}, ..., a_{C_k}$ from table
      $\mathbf{LT}_C$ for selected rows in the previous step.
4     Set sources patterns of new production rule on the basis of minimal
      values $e_1, ..., e_k$ of attributes that were found in the previous step.
5     Set the target pattern of new production, i.e., concept $C$ with the
      value $e$.
6     return $(C_1 \geq e_1) \wedge ... \wedge (C_k \geq e_k) \Rightarrow (C \geq e)$
7 end
```

5.5 Relation of Production Rules with DRSA

In 1996 Professor Greco, Professor Matarazzo and Professor Słowiński proposed a generalization of rough set theory for the need of multi-criteria decision problems (see, *e.g.*, [308, 309, 310, 311]). The main idea of this generalization is replacing the indiscernibility relation with the dominance relation. This approach is known under the *Dominance-based Rough Set Approach* (DRSA). In DRSA it is assumed that the values of all attributes of a given decision table (together with the decision attribute) are organized in a preferential way, that is, they are the so-called *criteria*. This means that for each attribute a from a given decision table a two-argument outranking relation is defined on the set of objects from this table, and at the same time a pair (x, y) belongs to this relation if the object x is *at least as good as* object y with regard to the criterion a. Using the outranking relation the dominance relation of one object on another is defined with respect to the established criteria set (attributes). Namely, the object x dominates object y with respect to the criteria set B (attributes), when x outranks y with respect to all criteria from B.

In DRSA it is possible to construct specific decision rules which is are called *dominance-based decision rules* (see, *e.g.*, [312]). Elementary conditions in the conditional part of these rules represent the statement that the object satisfy a criterion a (attribute) at least (or at most) as good as a certain established value of attribute a. Moreover, decision parts of the rules indicate that the object belongs to at least (or at most) to a given decision class.

Let us assume that there is given the layer table $\mathbf{LT}_C$ for the concept C which depends on concepts C_1,...,C_k in a certain ontology. It is easy to notice that each production rule (see Section 5.3) computed for the table $\mathbf{LT}_C$ by Algorithm 5.1 is a specific case of dominance-based rule (in the DRSA approach) established for the table $\mathbf{LT}_C$. Namely, it is such a case of dominance-based rule that when in the dominance-based rule predecessor, we consider the expression "the object is at least as good as" in relation the concept layers C_1,...,C_k represented by conditional attributes of $\mathbf{LT}_C$ table, and in the dominance-based rule successor "the object belongs at least to a given decision class" where decision classes are layers of the concept C. Moreover, in the rule predecessor of such a dominance-based rule there occur all conditional attributes.

On account of that, to establish production rules we may successfully apply algorithms known from literature for the induction of rules in the DRSA approach (see, *e.g.*, [312, 313, 314, 315]).

Using this approach to establish production rules, it should be remembered that the calculated dominance-based rules do not often have all conditional attributes in the predecessor. Meanwhile, according to the definition, each production rule has all conditional attributes from the table $\mathbf{LT}_C$ in the predecessor. However, with an appropriate interpretation each dominance-based rule may be treated as a production rule. It is enough to add to the predecessor all descriptors corresponding to the rest of conditional attributes; and at the same time each of the new descriptors must be constructed in such a way as to make all tested objects match it. This effect may be achieved by placing in the descriptor the attribute value representing the smallest preference possible.

5.6 Productions

Although a single production rule may be used as a classifier for the concept appearing in a rule predecessor, it is not yet a complete classifier, i.e., allowing to classify all objects belonging to an approximated concept, and not only those which match concepts of a rule predecessor. Therefore, in practice production rules are grouped into the so called *productions* (see, *e.g.*, [77, 89, 172, 188, 189, 190, 191, 192, 193]), i.e., production rule collections, in a way that to each production there belong rules having patterns for the same concepts in a predecessor and successor, but responding to their different layers.

Definition 18 (*A production*)**.** *Let us assume that:*

- $\mathbf{A} = (U, A)$ *is a given information system of unstructured objects of the fixed type,*
- C *is a concept, dependent in some ontology on concepts* C_1,...,C_k, *where* $C \subseteq U$ *and* $C_i \subseteq U$, *for* $i = 1, ..., k$,
- $\mathbf{T}_i = (U, A_i, d_{C_i})$ *is a decision table constructed for approximation of the concept* C_i *such that* $A_i \subseteq A$ *for* $i = 1..., k$ *and* d_{C_i} *is a decision attribute which values describe the membership of objects from* U *to the concept* C_i,
- $\mu_{C_i}^{E_i}$ *is a stratifying classifier for the concept* C_i *constructed on the basis the table* $\mathbf{T}_i$, *which classifies objects from the set* U *to layers, denoted be labels from the set* E_i, *for* $i = 1, ..., k$,

– $\mathbf{A}_C = (U_C, A_C)$ *is a concept approximation table for the concept* C *using stratifying classifiers* $\mu_{C_i}^{E_i}$*, for* $i = 1, ..., k$,
– μ_C^E *is a stratifying classifier for the concept* C*, which classifies objects from the set* U_C *to layers, denoted be labels from the set* E.

1. *A family of production rules* $P = \{r_1, ..., r_m\}$ *is a production if and only if for any pair of production rules* $r_i, r_j \in P$ *such that* $r_i = (p_1 \wedge ... \wedge p_k \Rightarrow p)$ *and* $r_j = (q_1 \wedge ... \wedge q_k \Rightarrow q)$ *and* $i < j$ *the following two conditions are satisfied:*
   – $|p_i|_{PEC(\mu_{C_i}^{E_i})} \subseteq |q_i|_{PEC(\mu_{C_i}^{E_i})}$ *for* $i = 1, ..., k$,
   – $|p|_{PEC(\mu_C^E)} \subseteq |q|_{PEC(\mu_C^E)}$.
2. *The domain of a given production is a sum of all domains of its production rules.*

Bellow, we present an example of production.

Example 17. In Fig. 22 we present three production rules constructed for some concepts C_1, C_2 and C approximated by three linearly ordered layers *"certainly NO"*, *"rather NO"*, *"possibly NO"*, *"possibly YES"*, *"rather YES"* and *"certainly YES"*. This collection of production rules is an exemplary production for concepts C_1, C_2 and C. Moreover, production rules from Fig. 22 have the following interpretation:

1. if inclusion degree to a concept C_1 is at least "*rather YES*" and to concept C_2 at least "*certainly YES*" then the inclusion degree to a concept C is at least "*certainly YES*";
2. if the inclusion degree to a concept C_1 is at least "*possibly YES*" and to a concept C_2 at least "*rather YES*" then the inclusion degree to a concept C is at least "*rather YES*";
3. if the inclusion degree to a concept C_1 is at least "*possibly YES*" and to a concept C_2 at least "*possibly YES*" then the inclusion degree to a concept C is at least "*possibly YES*". □

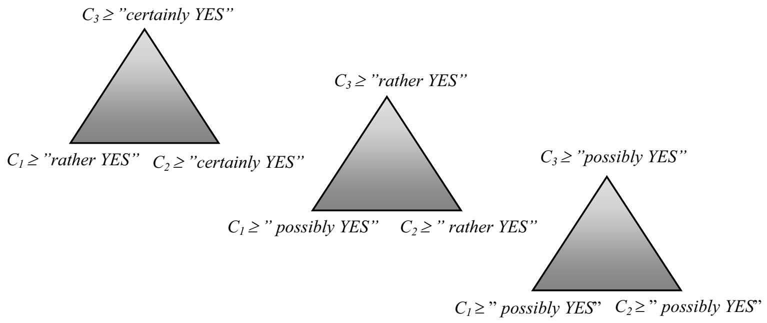

$C_3 \geq$ "certainly YES"
$C_1 \geq$ "rather YES"
$C_2 \geq$ "certainly YES"
$C_3 \geq$ "rather YES"
$C_1 \geq$ " possibly YES"
$C_2 \geq$ " rather YES"
$C_3 \geq$ "possibly YES"
$C_1 \geq$ " possibly YES"
$C_2 \geq$ " possibly YES"

Fig. 22. The example of production as a collection of three production rules

Algorithm 5.2. Classifying objects by production

```
Input: Tested object $u$ and production $P$
Output: The membership of the object $u$ to the concept $C$ ($YES$ or $NO$)
1  begin
2      Select a complex concept $C$ from an ontology (e.g., Safe overtaking).
3      if the tested object should not be classified by a given production $P$
       extracted for the selected concept $C$ then
4          return HAS NOTHING TO DO WITH // The object does
               not satisfy the production guard
5      end
6      Find a rule from production $P$ that classifies object with the maximal
       degree to the target concept of rule
7      if such a rule of $P$ does not exist then
8          return I DO NOT KNOW
9      end
10     Generate a decision value for object from the degree extracted in the
       previous step
11     if the extracted degree is greater than fixed threshold (e.g., possibly
       YES) then
12         return $YES$ // the object is classified to $C$
13     else
14         return $NO$ // the object is not classified to $C$
15     end
16 end
```

In the case of production from Fig. 22 concept C is the target concept and C_1, C_2 are the source concepts.

Any production can be used as a classifier. The method of object classification based on production can be described as follows:

1. Preclassify object to the production domain.
2. Classify object by production.

We assume that for any production *a production guard* is given. Such a guard describes the production domain and is used in preclassification of tested objects. The production guard definition is usually based on the relation of constraints (see Section 4.10) and its usage consists in checking whether a given object satisfies the constraints, that is, if it belongs to the relation of constraints.

For example, let us assume that the production P is generated for the concept: *Is the vehicle overtaking safely?*. Then an object-vehicle u is classified by production P iff u is overtaking. Otherwise, it is returned a message *"HAS NOTHING TO DO WITH (OVERTAKING)"*.

Now, we can present an exemplary algorithm for classifying objects by production (see Algorithm 5.2).

It is worth noticing that for objects which went through preclassification positively, two cases should be distinguished: object classification through production and recognizing the object through production. Classification of object through production means that such a production rule is found in the production that the tested object matches all patterns of its predecessor. However, not classifying the object through production means that such a production rule is not found. There also exists a third possibility that the tested object is not recognized by production. It means that relying on production rules in production, it is neither possible to state whether the tested object is classified by production nor that it is not. This case concerns the situation when stratifying classifiers realizing in practice the production patterns in the production rule predecessor are not able to recognize a tested object. This difficulty may be greatly decreased or even removed by applying to production patterns defining such classifiers that always or almost always classify objects.

A questions arises whether Algorithm 5.2 is universal enough to serve not only classifying tested objects from a given concept approximation table $\mathbf{A}_C$ (see Definition 15), but also to classify objects belonging to the extension of this table. Algorithm 5.2 works on the basis of production P, which is a family of production rules generated for concept approximation table $\mathbf{A}_C$. The application of each production rule only requires computation of the values of conditional attributes of table $\mathbf{A}_C$. It is done with the use of stratifying classifiers which were generated for lower ontology level concepts C_1,...,C_k. These classifiers are based on decision rules, therefore they may effectively classify tested objects outside a given information system $\mathbf{A}$ (see Definition 15 and Section 2.8). It would seem that this property transfers to Algorithm 5.2 where tested objects are classified with the help of production rules. Unfortunately, although to classify tested objects outside table $\mathbf{A}_C$ production rules may be applied, it is quite natural that production rules classify tested objects incorrectly. It results from the fact that production rules were constructed on the basis of dependencies observed between the attribute values of table $\mathbf{A}_C$, whereas in the extension of this table these dependencies may not occur. Therefore, similarly to the case of classifiers based on decision rules, while using production rules to classify objects, arguments for and against the membership of the tested object to a given concept should be taken into consideration. Obviously, such duality of arguments leads to conflicts in classifying tested objects and these conflicts must be appropriately resolved. In practice, it means that apart from production rules classifying a tested object to a given concept C, also production rules classifying tested objects to the complement of concept C should be taken into consideration. The complement of a given concept may be treated as a separate concept $C' = U \setminus C$. Production rules may also be generated for concept C' with the help of Algorithm 5.1. It requires, however, a suitable redefining layers of concepts C_1,...,C_k and sometimes using another ontology to approximate C'. As a result we obtain table $\mathbf{A}_{C'}$, which may serve generating production rules. Having production P_C generated for concept C and production $P_{C'}$ for concept C', the Algorithm 5.2 may be modified in such a way as to be able to resolve conflicts which may occur between production rules from P_C and $P_{C'}$. In this paper, we propose the following way of resolving these conflicts. If p_C and $p_{C'}$ are production rules chosen by Algorithm 5.2 from productions P_C and $P_{C'}$ respectively, then the tested object is classified to concept C only when the degree of certainty of classification by p_C is higher than the degree of certainty of classification by $p_{C'}$. Otherwise, the tested object is classified to C'.

5.7 Approximate Reasoning Schemes

Both productions and production rules themselves are only constructed for the two adjacent levels of ontology. Therefore, in order to use the whole ontology fully there are constructed the so called *approximate reasoning schemes* which are hierarchical compositions of production rules (see, *e.g.*, [77, 89, 172, 188, 189, 190, 191, 192, 193]).

The synthesis of AR-scheme is carried out in a way that to a particular production rule r lying on a lower hierarchical level of AR-scheme under construction another production rule r' on a higher level may be attached. However, this may be done only if one of the concepts for which the pattern occurring in the predecessor of r' was constructed is the concept corresponding to the successor pattern of the rule r. Additionally, it is required that the pattern occurring in a rule predecessor from the higher level is a pattern superset occurring in a rule successor from the lower level (in the sense of inclusion object sets matching both patterns). To the two combined production rules some other production rules can be attached (from above, from below or from the side) and in this way a multilevel structure is made which is a composition of many production rules.

In Fig. 23 we have two productions. The target concept of the first production is C_5 and the target concept of the second production is the concept C_3. We select one production rule from the first production and one production rule from the second production. These production rules are composed and then a simple AR-scheme is obtained that can be treated as a new two-levels production rule. Notice, that the target pattern of lower production rule in this AR-scheme is the same as one of the source patterns from the higher production rule. In this case, the common pattern is described as follows: inclusion degree (of some pattern) to a concept C_3 is at least *"possibly YES"*.

In this way, we can compose AR-schemes into hierarchical and multilevel structures using productions constructed for various concepts. AR-scheme constructed in such a way can be used as a hierarchical classifier whose input is given by predecessors of production rules from the lowest part of AR-scheme hierarchy and the output is a successor of a rule from the highest part of the AR-scheme hierarchy.

In this paper, there are proposed two approaches for constructing AR-schemes. The first approach is based on determining productions for a given ontology with the use of available data sets. Next, on the basis of these productions many AR-schemes are arranged which classify objects to different patterns on different ontology levels. All these productions and AR-schemes are stored in memory and their modification and potential arrangement of new AR-schemes is possible.

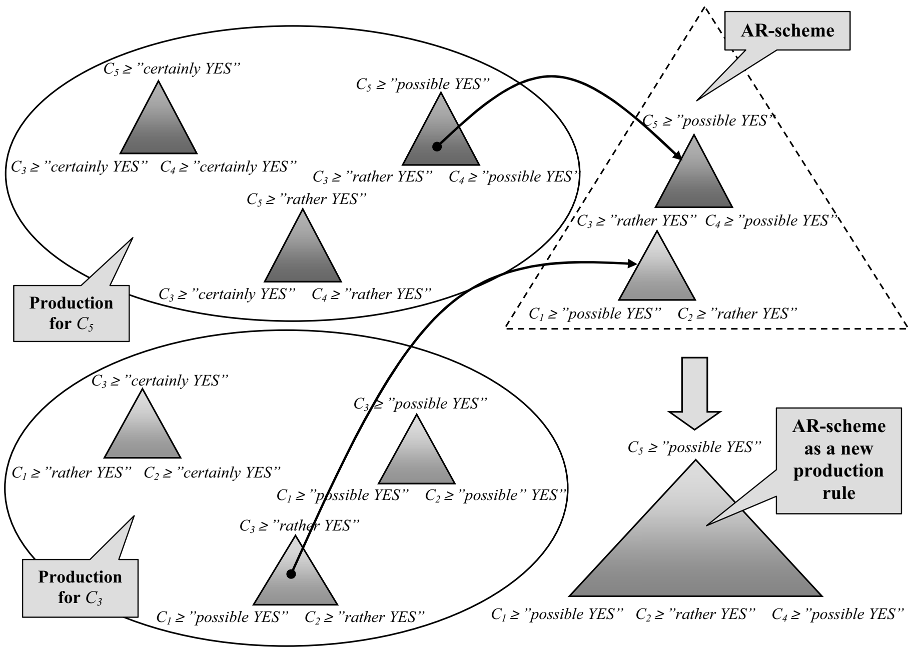

AR-scheme
$C_5 \geq$ "certainly YES"
$C_3 \geq$ "certainly YES" $C_4 \geq$ "certainly YES"
$C_5 \geq$ "possible YES"
$C_3 \geq$ "rather YES" $C_4 \geq$ "possible YES"
$C_5 \geq$ "rather YES"
$C_3 \geq$ "certainly YES" $C_4 \geq$ "rather YES"
Production for C_5
$C_5 \geq$ "possible YES"
$C_3 \geq$ "rather YES" $C_4 \geq$ "possible YES"
$C_1 \geq$ "possible YES" $C_2 \geq$ "rather YES"
$C_3 \geq$ "certainly YES"
$C_1 \geq$ "rather YES" $C_2 \geq$ "certainly YES"
$C_3 \geq$ "possible YES"
$C_1 \geq$ "possible YES" $C_2 \geq$ "possible" YES"
$C_3 \geq$ "rather YES"
$C_1 \geq$ "possible YES" $C_2 \geq$ "rather YES"
Production for C_3
$C_5 \geq$ "possible YES"
AR-scheme as a new production rule
$C_1 \geq$ "possible YES" $C_2 \geq$ "rather YES" $C_4 \geq$ "possible YES"

Fig. 23. Synthesis of approximate reasoning scheme

Hence, if a certain tested object should be classified, it is necessary to search in memory an AR-scheme appropriate for it and use it to classify the object. This approach enables steering the object classification depending on the expected certainty degree of the obtained classification. The drawback of this approach is a need of a large memory with a quick access to production and AR-schemes storage.

The second approach is based on a dynamic construction of AR-schemes. It is realized in a way that only during tested object classification itself, having been given different productions, an appropriate AR-scheme for classifying this particular object is built. Hence, this approach does not require so much memory as the previous approach. However, to its application we need the method of production method selection in dynamic construction of AR-schemes for tested object classification. A certain proposal of such a method is given by Algorithm 5.2 which suggests selecting from production such a production rule that recognizes the object, i.e., the object matches all patterns from the rule predecessor and production pattern from the successor of such a rule is based on a possibly highest layer, i.e., such a rule classifies the object possibly in the most certain way.

However, similarly to the case of a single production rule one AR-scheme is not yet a full classifier. That is why in practice there are many AR-schemes constructed for a particular concept which approximate different layers or concept regions. For example, on the basis of two productions from Fig. 23 three AR-schemes may be created which we show in Fig. 24. Each of these schemes is a classifier for one of production patterns constructed for the concept C_5.

The possibility of creating many AR-schemes for one concept is of a great practical significance, because tested objects may be classified to different production patterns which enable to capture the certainty degree with regard to membership of the tested object to the concept. For example, let us assume that we constructed five AR-schemes for the concept C_{SD} (safe driving) corresponding to production patterns: $C_{SD} \geq$ "*certainly YES*", $C_{SD} \geq$ "*rather YES*", $C_{SD} \geq$ "*possibly YES*", $C_{SD} \geq$ "*possibly NO*" and $C_{SD} \geq$ "*rather NO*". If a certain tested object is not classified by the AR-scheme constructed for the $C_{SD} \geq$ "*certainly YES*" pattern, then we cannot conclude with certainty that this object is driving safely. However, what also should be checked is the fact if this object is not classified by the ARscheme constructed for the $C_{SD} \geq$ "*rather YES*" pattern (then we may conclude that the object rather drives safely) or by the AR-scheme constructed by the $C_{SD} \geq$ "*possibly YES*" pattern (which means that the vehicle perhaps drives safely). Only if the tested object is not classified by any of these three AR-schemes, we may conclude that the tested object is not going safely. Then the question arises, how dangerously

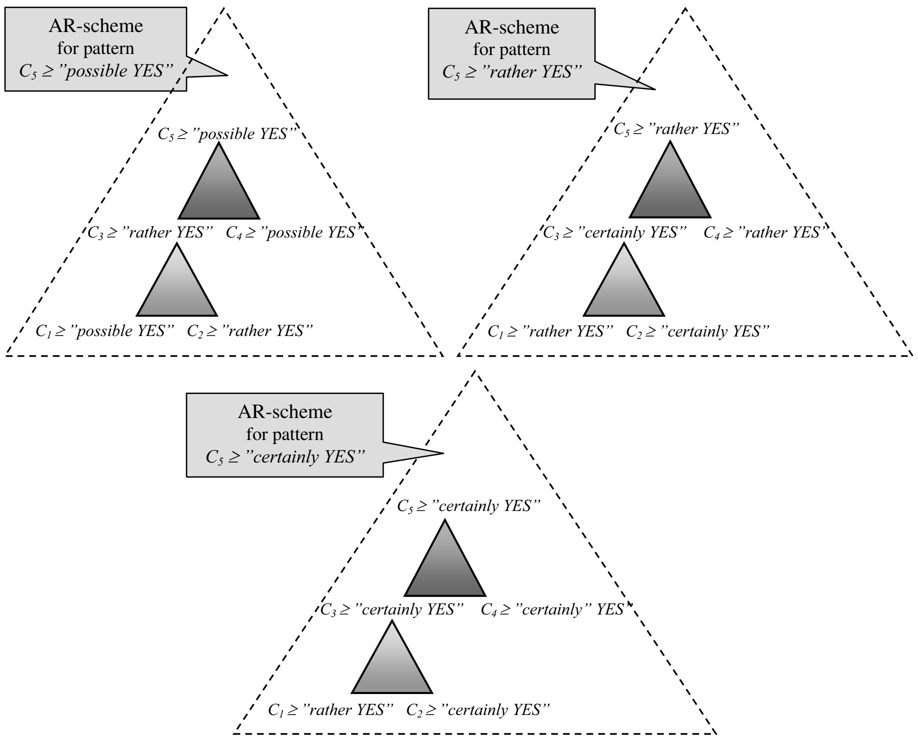


Fig. 24. Three approximate reasoning schemes for concept C_5

that object is behaving? To solve this it should be checked if the object is classified by the AR-scheme constructed for the $C_{SD} \geq$ "*possibly NO*" and then $C_{SD} \geq$ "*rather NO*" pattern. Only if none of these AR-schemes classifies the object, we may conclude that the tested object certainly does not go safely.

It is worth noticing that similarly to the case of production rule, two cases should be distinguished here: object classification by the AR-scheme and object recognition by the AR-scheme. Object classification by the AR-scheme means that the tested object belongs to all patterns lying at the bottom of the AR-scheme and this object is classified to the pattern lying at the top of the AR-scheme. However, not classifying the object means that the tested object does not belong to at least one of the patterns lying at the bottom of the AR-scheme. There is a third possibility that the tested object is not recognized by the AR-scheme. This means that, relying on a given AR-scheme, it is not possible to state that the tested object belongs to the pattern lying at the top of the AR-scheme. This case concerns the situation when stratifying classifiers executing, in practice, patterns lying at the bottom of the AR-scheme are not able to recognize the tested object. In such a situation with regard to classifying the tested object, there are two ways of procedure. Firstly, it may be acknowledged that the tested object cannot be classified using available AR-schemes. However, this approach frequently causes that the number unclassified objects is too large. Therefore, in practice the other approach is applied which consists in trying to classify the tested object with the AR-schemes classifying objects to patterns representing smaller certainty of the concept belonging, counting on the fact that such AR-schemes have a greater extension. The drawback of this approach is, however, the fact that a false resulting in decrease of the certainty of the tested object's membership to the concept is possible. This difficulty may be greatly diminished or even removed by applying, in the production pattern, such classifiers that always or almost always classify objects.

It is worth noticing that similarly to the case of production rules, in the case of using AR-schemes to construct classifiers, arguments for and against the membership of the tested object to a given concept should be taken into consideration. Thus, the obtained classifier will be able to serve effective classification not only of tested objects from a given concept approximation table $\mathbf{A}_C$ (see Definition 15) but also to classify objects belonging to the extension of this table. In practice it means that apart from AR-schemes which classify the tested object to a given concept, also AR-schemes which classify tested objects to the complement of this concept should be taken into consideration. Obviously, conflicts occurring at this point in classification of tested objects should be appropriately resolved, for instance like in the case of conflicts between production rules (see Section 5.6).

5.8 Experiments with Data

To verify effectiveness of classifiers based on AR schemes, we have implemented our algorithms in the AS-lib programming library. This is an extension of the

RSES-lib programming library creating the computational kernel of the RSES system (see Section 2).

The experiments have been performed on the data set obtained from the road simulator (see Appendix A). Data set consists of 18101 objects generated by the road simulator. We have applied the train and test method. The data set was randomly divided into two parts: training and test ones (50% + 50%). In order to determine the standard deviation of the obtained results each experiment was repeated for 10 random divisions of the whole data set.

In our experiments, we compared the quality of two classifiers: RS and ARS. For inducing RS we use RSES system generating the set of decision rules by algorithm LEM2 (see Section 2.4) that are next used for classifying situations from testing data. ARS is based on AR schemes.

During ARS classifier construction, in order to approximate concepts occurring in ontology we used the LEM2 algorithm (see Section 2.4).

For production rule construction we used the expert method of stratifying classifier construction (see Section 3.2). However, to classify objects using the ARS classifier we used the method of dynamic construction of the AR-schemes for specific tested objects (see Section 5.7).

We compared RS and ARS classifiers using the accuracy, the coverage, the accuracy for positive examples (the sensitivity or the true positive rate), the accuracy for negative examples (the specificity or the true negative rate), the coverage for positive examples and the coverage for negative examples, the learning time and the rule set size (see Section 2.9).

Table 2 shows the results of the considered classification algorithms for the concept *Is the vehicle driving safely?* (see Fig. 6). Together with the results we present a standard deviation of the obtained results.

One can see that accuracy of algorithm ARS for the decision class *NO* is higher than the accuracy of the algorithm RS for analyzed data set. The decision

Table 2. Results of experiments for the concept: *Is the vehicle driving safely?*

Decision class	Method	Accuracy	Coverage	Real accuracy
YES	RS	0.977 ± 0.001	0.948 ± 0.003	0.926 ± 0.003
	ARS	0.967 ± 0.001	0.948 ± 0.003	0.918 ± 0.003
NO	RS	0.618 ± 0.031	0.707 ± 0.010	0.436 ± 0.021
	ARS	0.954 ± 0.016	0.733 ± 0.018	0.699 ± 0.020
All classes	RS	0.963 ± 0.001	0.935 ± 0.003	0.901 ± 0.003
(YES + NO)	ARS	0.967 ± 0.001	0.937 ± 0.004	0.906 ± 0.004

Table 3. Learning time and the rule set size for concept: *Is the vehicle driving safely?*

Method	Learning time	Rule set size
RS	488 ± 21 seconds	975 ± 28
ARS	33 ± 1 second	174 ± 3

class *NO* is smaller that the class *YES*. It represents atypical cases in whose recognition we are most interested in (dangerous driving a vehicle on a highway).

Table 3 shows the learning time and the number of decision rules induced for the considered classifiers. In the case of the algorithm ARS we present the average number of decision rules over all concepts from the relationship diagram (see Fig. 6).

One can see that the learning time for ARS is much shorter than for RS and the average number of decision rules (over all concepts from the relationship diagram) for ARS algorithm is much lower than the number of decision rules induced for RS.

The experiments showed that classification quality obtained through classifiers based on AR-schemes is higher than classification quality obtained through traditional classifiers based on decision rules (especially in the case of the class *NO*). Apart from that the time spent on classifier construction based on AR-schemes is shorter than when constructing classical rule classifiers. Also, the structure of a single rule classifier (inside the ARS classifier) is less complicated than the structure of RS classifier (a considerably smaller average number of decision rules). It is worth noticing that the the performance of the ARS classifier is much more stable than the RS classifier because of the differences in data in samples supplied for learning (*e.g.*, to change the simulation scenario).

6 Behavioral Pattern Identification

An efficient complex dynamical systems monitoring very often requires the identification of the so-called *behavioral patterns* or a specific type of such patterns called *high-risk patterns* or *emergent patterns* (see, *e.g.*, [93, 99, 100, 132, 138, 139, 140, 141, 142, 143, 144, 173, 174, 175, 176]). They are complex concepts concerning dynamic properties of complex objects, dependent on time and space and expressed in a natural language. Examples of behavioral patterns may be *overtaking on a road*, *behavior of a patient faced with a serious life threat*, *ineffective behavior of robot team.* These types of concepts are much more difficult to approximate than complex concepts whose approximation does not require following object changes over time and may be defined for unstructured or structured objects. Identification of some behavioral patterns can be very important for recognition or prediction of behavior of a complex dynamical system, *e.g.*, some behavioral patterns correspond to undesirable behaviors of complex objects. In this case we call such behavioral patterns as *risk patterns* and we need some tools for identifying them. If in the current situation some risk patterns are identified, then the control object (a driver of the vehicle, a medicine doctor, a pilot of the aircraft, etc.) can use this information to adjust selected parameters to obtain the desirable behavior of the complex dynamical system. This can make it possible to overcome dangerous or uncomfortable situations. For example, if some behavior of a vehicle that cause a danger on the road is identified, we can try to change its behavior by using some suitable means such as road traffic signalling, radio message or police patrol intervention. Another

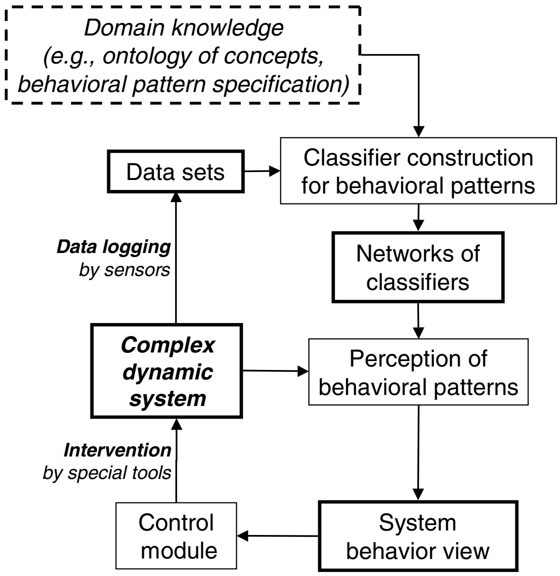

Domain knowledge (e.g., ontology of concepts, behavioral pattern specification)
Data sets
Classifier construction for behavioral patterns
Data logging by sensors
Networks of classifiers
Complex dynamic system
Perception of behavioral patterns
Intervention by special tools
Control module
System behavior view

Fig. 25. Complex dynamical systems monitoring using behavioral patterns

example can be taken from medical practice. A very important element of the treatment of the infants with respiratory failure is appropriate assessment of the risk of death. The appropriate assessment of this risk leads to the decision of particular method and level of treatment. Therefore, if some complex behavior of an infant that causes a danger of death is identified, we can try to change its behavior by using some other methods of treatment (may be more radical) in order to avoid the infant's death (see Section 6.26).

In the Fig. 25 a scheme of complex dynamical system monitoring with the help of behavioral patterns is presented. This monitoring takes place in the following way. At the beginning, as a result of complex dynamical system observation, there are registered data sets describing changing over time parameter values of complex objects occurring in the system under observation. For a given complex dynamic system domain knowledge is gathered concerning, among others, complex behaviors of objects occurring in this system. Next, classifier nets are constructed on the basis of this knowledge and gathered data sets which enable perception of these patterns' behaviors whose detection is crucial for the correct functioning of the complex dynamical system. Identification of such patterns enable to find out important facts about the current system situation. This knowledge may be used by a control module which may perform a sequence of intervening actions aiming at restoring or maintaining the system in a safe, correct or convenient condition. Moreover, during complex dynamical systems monitoring data sets may still be collected. On the basis of these data sets a classifier structure identifying behavioral patterns is updated. This enable a certain type of adaptation of applied classifiers.

In this section a methodology of complex object's behavior monitoring is proposed which is to be used for approximating behavioral patterns on the basis of data sets and domain knowledge.

6.1 Temporal Information System

The prediction of behavioral patterns of a complex object evaluated over time is usually based on some historical knowledge representation used to store information about changes in relevant futures or parameters. This information is usually represented as a data set and has to be collected during long-term observation of a complex dynamical system (see, *e.g.*, [173, 174, 175, 176, 316, 318]). For example, in the case of road traffic, we associate the object-vehicle parameters with the readouts of different measuring devices or technical equipment placed inside the vehicle or in the outside environment (*e.g.*, alongside the road, in a helicopter observing the situation on the road, in a traffic patrol vehicle). Many monitoring devices serve as informative sensors such as Global Positioning System (GPS), laser scanners, thermometers, range finders, digital cameras, radar, image and sound converters (see, *e.g.*, [97, 153]). Hence, many vehicle features serve as models of physical sensors. Here are some exemplary sensors: location, speed, current acceleration or deceleration, visibility, humidity (slipperiness) of the road. By analogy to this example, many features of complex objects are often dubbed sensors. It is worth mentioning, that in the case of the treatment of infants with respiratory failure, we associate the object parameters (sensors) mainly with values of arterial blood gases measurements and the X-ray lung examination.

Data sets used for complex object information storage occurring in a given complex dynamical system may be represented using information systems (see, *e.g.*, [318]). This representation is based on representing individual complex objects by object (rows) of information system and information system attributes represent the properties of these objects. Because in a complex dynamical system there may occur many different complex objects, the storing of information about individual complex object identifiers is necessary. This information may be represented by the distinguished information system attribute which we mark by a_{id}. For convenience of the further discussion (see Algorithm 6.2) we assume that the set of values of the a_{id} attribute is linearly ordered. Therefore, the a_{id} attribute must be enriched by the relation ordering the set of values of this attribute in a linear order. Apart from that, it should be remembered that the complex objects occurring in complex dynamical systems change over time and their properties (object states) should be registered at different time instants (in other words time points). Hence, it is also necessary to store together with the information about a given object an identifier of time in which these properties are registered. This information may also be represented by the distinguished information system attribute which we mark as a_t. Because we assume that the identifiers of a time point are linearly ordered, then attribute a_t must be enriched by a relation ordering the set of values of this attribute in a linear order.

Hence, in order to represent complex object states observed in complex dynamical systems, the standard concept of information system requires extension. Therefore, we define *a temporal information system* [318].

Definition 19 (*A temporal information system*)**.**

1. *A temporal information system is a six-element tuple:*

$$\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t}),\ \textit{where:}$$

   (a) (U, A) *is an information system,*
   (b) a_{id}, a_t *are distinguished attributes from the set* A*,*
   (c) $\leq_{a_{id}}$ *is a relation of linear order on the set* $V_{a_{id}}$*,*
   (d) $\leq_{a_t}$ *is a relation of linear order on the set* V_{a_t}*.*
2. *About an object* $u \in U$ *we say that it represents the current parameters of the complex object with identifier* $a_{id}(u)$ *at time point* $a_t(u)$ *in the temporal information system* $\mathbf{T}$*.*
3. *About an object* $u_1 \in U$ *we say that it precedes an object* $u_2 \in U$ *in the temporal information system* $\mathbf{T}$ *if and only if*

$$u_1 \neq u_2 \ \wedge\ a_{id}(u_1) = a_{id}(u_2) \ \wedge\ a_t(u_1) \leq_{a_t} a_t(u_2).$$

4. *About an object* $u_2 \in U$ *we say that it follows an object* $u_1 \in U$ *in the temporal information system* $\mathbf{T}$ *if and only if* u_1 *precedes* u_2*.*
5. *About an object* $u \in U$ *we say that it is situated between objects* $u_1, u_2 \in U$ *in the temporal information system* $\mathbf{T}$ *if and only if* u_1 *precedes* u *and* u *precedes* u_2*.*

A typical example of a temporal information system is an information system whose objects represent vehicles' states at different instants of their observation.

Example 18. Let us have temporal information system $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ whose objects represent vehicles' states at different instants of their observation. Attributes from the set A describe sensor parameters of the vehicle at individual instants (*e.g.*, speed, location, lane, etc.). The distinguished a_{id} attribute is a unique number identifier of each vehicle, registered in the system $\mathbf{T}$. The a_t attribute represents the number of time units (*e.g.*, seconds) which have elapsed since the starting moment of all vehicles' observation. However, the relations $\leq_{a_{id}}$ and $\leq_{a_t}$ are common relations $\leq$ on the set of natural numbers. □

6.2 Representing Spatial Properties Using Concepts

In the presented approach the first step to identify the behavior of complex objects changing over time is representing and recognizing spatial properties of complex objects, that is, those which concern a certain chosen time point and their recognition does not require observing complex object changes over time. One of the most popular ways to represent these properties is representing them using concepts. Each concept introduces the partitioning the sets of objects into two classes, that is, the class of these objects which belong to the concept and at the same time satisfy the property connected with the concept and the class of objects not belonging to the concept and at the same time not satisfying the property connected with the concept. If complex objects changing over time are represented using temporal information systems, then the concepts representing these objects' properties may be defined using attributes available in this system. The language of defining such concepts may be for example the language $GDL(\mathbf{T})$ (see Definition 5) where $\mathbf{T}$ is a temporal information system. Using this language spatial properties of complex objects may be observed at single time points. They may be for example, such concepts as: *low vehicle speed*, *high body temperature*, *considerable turn or dangerous inclination of a robot.*

Another language allowing to define properties of complex objects is *a language of elementary changes of object parameters* using information about how at a given time instant the values of the elementary parameters of the complex object changed in relation to the previous observation of this object (see [88, 173, 174, 175, 176, 178]). This property defining language is very useful when we wish to observe complex object parameters in relation to their previous observation. Examples of such properties may be: *increasing or decreasing the speed of the vehicle*, *moving the vehicle towards the right lane*, *the increasing the patient's body temperature.*

However, the use of the two above mentioned languages for defining properties of complex objects is possible only when the concepts being defined can be defined using formulas which use attribute values representing the current or previous value of the complex object's parameter. In practice, formulas of this type may be defined by experts on the basis of domain knowledge. However, an expert is often not able to give such an accurate definition of the concept. For example, the concept expressed using an expert's statement that *the vehicle speed is low* is difficult to be described without additional clues using a formula of the language GDL based on sensor attributes, although intuitively the dependence of this concept on the sensor attributes does not raise any doubt. Similarly, an expert's statement that *the patient's body temperature has fallen slightly since the last observation* is difficult to be formally described without additional clues, using the language of elementary changes of the complex object parameters (in this case the complex object is a treated patient). Meanwhile, in everyday life we often use such statements. Therefore, in the general case describing concepts concerning complex object properties requires the approximation of these concepts with the help of classifiers. This approximation may take place on the basis of a decision table whose conditional attributes are attribute arrangement from a given information system, while the decision attribute values given by the expert (on the basis of domain knowledge) describe the membership of the objects of the table under construction to the concept being approximated. The classifier constructed for such a table allow to test the membership of any object to the concept being approximated.

6.3 Temporal Information System Based on Concepts

If we decide to represent complex object properties using concepts, then a specific type of temporal information system is necessary which we call *a temporal information system based on concepts.*

Definition 20 (*A temporal information systems based on concepts*)**.** *A temporal information system* $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ *we call a temporal information system based on concepts (c-temporal information system or a c-system), if all attributes from the set* A *apart from the* a_{id} *and* a_t *attributes are attributes representing concepts determined in the set of objects* U*.*

Each attribute of c-system (apart from the a_{id} and a_t attributes) is then a binary attribute (taking two values). In this paper, we assume that they are 1 and 0 values, with 1 symbolizing the membership of the objects to the concept and 0 symbolizing the membership of the object to the concept complement.

Data sets gathered for complex dynamical systems and represented using temporal information systems usually contain continuous attributes, that is, ones with a large number of values which we often associate with different sensor indications. Therefore, if we wish to use c-systems for learning complex behavior of objects changing over time, then at the beginning of the learning process a c-system must be constructed on the basis of the available temporal information system. In order to do this a family of concepts must be defined which replaces all attributes (apart from the a_{id} and a_t attributes) of the original temporal information system. It is also necessary to construct a family of classifiers which approximate concepts from the defined family of concepts. These classifiers serve as replacements of the attributes of the input system with the c-system attributes. Such an operation are called the *c-transformation.*

Definition 21. *Let us assume that:*

- $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ *is a temporal information system,*
- $C_1, ..., C_k$ *is a family of concepts defined on* U*,*
- $\mu_1, ..., \mu_k$ *are a family of classifiers approximating concepts* $C_1, ..., C_k$ *based on the chosen attributes from set* $A \setminus \{a_{id}, a_t\}$*.*

1. *An operation of changing the system* $\mathbf{T}$ *to a c-system*

$$\mathbf{T}_c = (U, A_c, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$$

   is called a c-transformation of the system $\mathbf{T}$*, if* $A_c = \{a_{id}, a_t, c_1, ..., c_k\}$ *and for any* $u \in U : c_i(u) = \mu_i(u)$*, for* $i = 1, ..., k$*;*
2. *The C-system* $\mathbf{T}_c$ *is called a result of c-transformation of the system* $\mathbf{T}$*.*

Now, we present the c-transformation algorithm for the temporal information system (see Algorithm 6.1). Performance of this algorithm is based on constructing a new information system which apart from attributes a_{id} and a_t has all the attributes based on the previously defined concepts.

Algorithm 6.1. C-transformation

```
Input:
  1. temporal information system $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ such that
     $U = \{u_1, ..., u_n\}$,
  2. family of concepts $C_1, ..., C_k$ defined in the set $U$,
  3. family of classifiers $\mu_1, ..., \mu_k$ approximating concepts $C_1, ..., C_k$ on the
     basis of attributes from the set $A \setminus \{a_{id}, a_t\}$.

Output: The C-system $\mathbf{T}_c = (U, A_c, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ such that
        $A_c = \{a_{id}, a_t, c_1, ..., c_k\}$, where attributes $c_1, ..., c_k$ represent
        concepts $C_1, ..., C_k$
1  begin
2      Create an empty information system $\mathbf{T}_c$ which has attributes
           $a_{id}, a_t, c_1, ..., c_k$ where attributes $a_{id}$ and $a_t$ are of the identical
           type as their counterparts in system $\mathbf{T}$ and attributes $c_1, ..., c_k$ are
           binary attributes // $\mathbf{T}_c$ is without any objects for the
           time being
3      for $i := 1$ to $n$ do
4          Create an empty list of values $L$.
5          Add $a_{id}(u_i)$ to the list $L$.
6          Add $a_t(u_i)$ to the list $L$.
7          for $j = 1$ to $k$ do
8              Add $\mu_j(u_i)$ to the list $L$.
9          end
10         Add new object represented by values from $L$ to the system $\mathbf{T}_c$.
11     end
12     return $\mathbf{T}_c$
13 end
```

With the assumption that each of the classifiers $\mu_1, ..., \mu_k$ can classify an object within the time of order $O(C)$, where C is a certain constant, then the time complexity of the above algorithm is of order $O(n \cdot k)$, where $n = card(U)$ and k is the number of concepts used for constructing the attributes.

Example 19. Let us take into consideration the temporal information system such as the one in Example 18. In such a system there may occur many continuous attributes like: *the speed of the vehicle*, *the location of the vehicle with regard to the crossroads*, *the location of the vehicle with regard to the left and right lane*, *visibility* and others. Therefore, this system, before being used in order to approximate temporal concepts, requires c-transformation which has to be executed on the basis of the established concepts. Also, classifiers for these concepts which were constructed earlier with the use of attributes from a given system are necessary. They may be, for example, the following concepts:

1. low (average, high) vehicle speed (approximation using the attribute: speed),
2. increasing (decreasing, maintaining) the vehicle speed in relation to the previous time point (approximation using attributes: speed and speed in the previous time point),
3. high (average, low) distance from the crossroads (approximation using the attribute: distance from the crossroads),
4. driving in the right (left) lane (approximation using the attribute: the location of the vehicle with regard to the left and right lane),
5. small movement of the vehicle towards the left (right) lane (approximation using attributes: the location of the vehicle with regard to the left and right lane and the location of the vehicle with regard to the left and right lane in the previous time point),
6. location of the vehicle at the crossroads (symbolic attribute moved from the initial system),
7. good (moderate, bad) visibility on the road,
8. high humidity (low humidity, lack of humidity) of the road.

After performing the c-transformation, the temporal information system from Example 18 is already a c-temporal information system, that is, apart from a_{id} and a_t attributes all of its attributes are binary ones representing concepts. $\square$

Let us notice that the concepts applied during the c-transformation are usually constructed using discretization of chosen continuous attributes of a given information system performed manually by the expert. Obviously, this discretization may also be performed with the use of automatic methods (see Section 2.2).

6.4 Time Windows

The concepts concerning properties of unstructured complex objects at the current time point in relation to the previous time point are a way of representing very simple behaviors of the objects. However, the perception of more complex types of behavior requires the examination of behavior of complex objects over a longer period of time. This period is usually called the time window (see, *e.g.*, [173, 174, 175, 176, 316, 318]), which is to be understood as a sequence of objects of a given temporal information system registered for the established complex object starting from the established time point over the established period of time or as long as the expected number of time points are obtained. Therefore, learning to recognize complex types of behavior of complex objects with use of gathered data as well as the further use of learned classifiers to identify the types of behavior of complex objects, requires working out of the mechanisms of extraction of time windows from the data and their properties. That is, why we need the language of extraction of time windows from the c-system which we are about to define.

Definition 22 (*A language for extracting time windows*)**.** *Let* $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ *be a c-temporal information system and let* $\mathbb{Z}_2$ *be the set of integer numbers equal or greater than 2. A language for extracting time windows from* *system* $\mathbf{T}$ *(denoted by* $ETW(\mathbf{T})$ *or* ETW*-language, when* $\mathbf{T}$ *is fixed) is defined in the following way:*

- *the set* $AL_{ETW}(\mathbf{T}) = V_{a_{id}} \cup V_{a_t} \cup \mathbb{Z}_2 \cup \{\text{“,”}\}$ *is called an alphabet of the language* $ETW(\mathbf{T})$*,*
- *the set of atomic formulas of the language* $ETW(\mathbf{T})$ *is defined as a set of three-element tuples in the following form:* (i, b, s)*, where* $i \in V_{a_{id}}$*,* $b \in V_{a_t}$ *and* $s \in \mathbb{Z}_2$*.*

Now, we determine the semantics of the language $ETW(\mathbf{T})$. The language $ETW(\mathbf{T})$ formulas may be treated as the descriptions of object sequences occurring one after another in system $\mathbf{T}$.

Definition 23. *Let* $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ *be a c-temporal information system. A satisfiability of an atomic formula* $\phi = (i, b, s) \in ETW(\mathbf{T})$ *by an object* $u \in U$ *from* $\mathbf{T}$ *(denoted by* $u \models_{ETW(\mathbf{T})} \phi$*), is defined in the following way:*

$$u \models_{ETW(\mathbf{T})} (i, b, s) \Leftrightarrow$$

$$a_{id}(u) = i \ \wedge \ card(\{x \in U : x \ precedes \ u \wedge b \leq_{a_t} a_t(x)\}) < s.$$

Let us notice that an object $u \in U$ satisfies a formula $\phi = (i, b, s) \in ETW(\mathbf{T})$ iff the following two conditions are satisfied:

1. the identifier of the object u is equal i,
2. the number of objects registered since b to $a_t(u)$ is less than s.

Formulas of the language ETW describe sets of objects which we call *time windows.*

Definition 24 (*A time window*). *Let* $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ *be a c-temporal information system.*

1. *A time window in the c-temporal information system* $\mathbf{T}$ *is a set* $|\phi|_{ETW(\mathbf{T})}$*, where* $\phi \in ETW(\mathbf{T})$*.*
2. *The family of all time windows from the c-temporal information system* $\mathbf{T}$ *is denoted by* $TW(\mathbf{T})$*.*
3. *If* $W \in TW(\mathbf{T})$ *then the number* $card(W)$ *is called a length of time window* W *and is denoted by* $Length(W)$*.*
4. *The family of all time windows from the c-temporal information system* $\mathbf{T}$ *with length equals to* s *is denoted by* $TW(\mathbf{T}, s)$*.*

Because according to the definition of semantics of the language $ETW(\mathbf{T})$ the elements of each time window $W \in TW(\mathbf{T}, s)$ are linearly ordered by relation $\leq_{a_t}$, then each time window may be treated as an ordered sequence $W = (u_1, ..., u_s)$ of objects from set U. Additionally each i-th object of time window W we mark with $W[i]$, where $i \in \{1, ..., s\}$.

Here is an example of extraction of the time window from the c-temporal information system.

Example 20. Let us consider c-system $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ whose objects represent the states of vehicles at different time points. The attributes from set A describe concepts representing sensor properties of vehicle parameters at individual points (*e.g.*, high velocity, small acceleration, etc.). The distinguished attribute a_{id} is a unique identifier of each vehicle and attribute a_t represents the observation time registered in a given object of system $\mathbf{T}$. To simplify matters let us assume that the values of attributes a_t and a_{id} are natural numbers. Let us consider the vehicle with identifier 5 for which one hundred time points have been registered in the system from the time point with identifier 11 to the time point with identifier 109. For this vehicle we could, for example, isolate a time window defined by formula $(5, 20, 31)$ which represents the behavior of the vehicle from the time point marked 20 to the time point marked 50. $\square$

6.5 Temporal Concepts

More complex types of behavior of complex objects may be defined using time widows over complex concepts which we call temporal concepts. We assume that temporal concepts are specified by a human expert. Temporal concepts are usually used in queries about the status of some objects in a particular time window. Answers to such queries can be of the form *Yes*, *No* or *Does not concern*. For example, in the case of road traffic one can define complex concepts such as *Is a vehicle accelerating in the right lane?*, *Is a vehicle speed stable while changing lanes?*, or *Is the speed of a vehicle in the left lane stable?*.

Intuitively, each temporal concept (defined on the time window) depends on object properties observed at some time points. At the same time we mean both spatial properties, that is, properties registering the spatial value of the complex object parameter observed at the time point, *e.g.*, *the left driving lane of the vehicle*, *high speed of the vehicle*, *low the patient's body temperature*) as well as the properties describing elementary changes of complex object parameters in relation to the previous observation of this object (*e.g.*, *increasing or decreasing the speed of the vehicle*, *small move of the vehicle towards the right lane*, *the increasing the patient's body temperature*). Such simple concepts we call *elementary concepts*. Usually it is possible to provide the ontology which shows a dependence between a temporal concept and some elementary concepts. For example, the temporal concept *accelerating and changing lanes from right to left* depends on such elementary concepts as *high speed*, *low speed*, *increasing speed*, *decreasing speed*, *small move of the vehicle towards the left lane.*

6.6 Temporal Patterns

It would seem that temporal concepts as concepts on the higher hierarchical level of ontology may be approximated using elementary concepts which are on the lower ontology level (see Section 6.5). It is sufficient to build a concept approximation table for the approximated concept. However, during the construction of such a table we encounter a serious problem resulting from the difference in meaning (semantical difference) of objects being examples and counterexamples of concepts on both ontology levels. Therefore, concepts on the lower ontology level are defined for time points, while temporal concepts on the higher ontology level are determined on time windows. In other words the observations of objects at time points are examples and counterexamples for concepts on the lower ontology level. However, the objects which are examples and counterexamples for temporal concepts are the observations of complex objects registered over time windows, that is, sequences of object observations from time points. For this reason we cannot apply here the construction method of conditional attributes from Section 5.1 which is based on the language PEC, since that method required for the concepts existing on both ontology levels to concern the same type of objects.

That is why to define the attributes which approximate temporal concepts we need to introduce a different language which can make it possible to transfer the spatial properties of complex objects registered at time points onto the property level of complex objects over the time window. In this paper, for this purpose we propose *a language for definnig features of time windows.*

Definition 25 (*A language for definnig features of time windows*). *Let* $\mathbf{T} = (U,\ A,\ a_{id},\ \leq_{a_{id}},\ a_t,\ \leq_{a_t})$ *be a c-temporal information system. A language for definnig features of time windows of c-temporal information system* $\mathbf{T}$ *(denoted by* $FTW(\mathbf{T})$ *or* FTW*-language, when* $\mathbf{T}$ *is fixed) is defined in the following way:*

- *the set* $AL_{FTW}(\mathbf{T}) = (A \setminus \{a_{id}, a_t\}) \cup \{ExistsPoint,\ EachPoint,\ MajorityPoints,\ MinorityPoints,\ FirstPoint,\ LastPoint,\ OrderPoints\} \cup \{\neg, \vee, \wedge\}$ *is an alphabet of the language* $FTW(\mathbf{T})$,
- *for any* $a, b \in A \setminus \{a_{id}, a_t\}$ *expressions of the form* $ExistsPoint(a)$, $EachPoint(a)$, $MajorityPoints(a)$, $MinorityPoints(a)$, $FirstPoint(a)$, $LastPoint(a)$, $OrderPoints(a, b)$ *are atomic formulas of the language* $FTW(\mathbf{T})$.

Now, we determine the semantics of the language $FTW(\mathbf{T})$. The formulas of the language $FTW(\mathbf{T})$ may be treated as the descriptions of time windows in system $\mathbf{T}$. For example, the formula $ExistsPoint(a)$ is interpreted as the description of all those time windows of system $\mathbf{T}$ in which such an object u has been observed that $a(u) = 1$. Thus, we observed an object belonging to the concept represented by attribute a. Time windows may be described by different formulas, however, for formula ϕ to have sense in system $\mathbf{T}$, that is, to be semantically correct in the language $FTW(\mathbf{T})$, there has to exist at least one time window which is described by formula ϕ. For such a window we say that it *satisfies* formula ϕ.

Definition 26. *Let* $\mathbf{T} = (U,\ A,\ a_{id},\ \leq_{a_{id}},\ a_t,\ \leq_{a_t})$ *be a c-information system and let* s *be a length of time windows. The satisfiability of an atomic formula* $\phi \in FTW(\mathbf{T})$ *by a time window* $W \in TW(\mathbf{T}, s)$ *(denoted by* $W \models_{FTW(\mathbf{T})} \phi$*), is defined in the following way:*

1. $W \models_{FTW(\mathbf{T})} ExistsPoint(a) \Leftrightarrow$ *exists* $u \in W$ *such that* $a(u) = 1$,
2. $W \models_{FTW(\mathbf{T})} EachPoint(a) \Leftrightarrow$ *for any* $u \in W$ *is satisfied* $a(u) = 1$,

3. $W \models_{FTW(\mathbf{T})} MajorityPoints(a) \Leftrightarrow$

$$card(\{u \in W : a(u) = 1\}) > \lfloor (Length(W) - 1)/2 \rfloor,$$

4. $W \models_{FTW(\mathbf{T})} MinorityPoints(a) \Leftrightarrow$

$$card(\{u \in W : a(u) = 1\}) < \lceil (Length(W) - 1)/2 \rceil,$$

5. $W \models_{FTW(\mathbf{T})} FirstPoint(a) \Leftrightarrow a(W[1]) = 1,$
6. $W \models_{FTW(\mathbf{T})} LastPoint(a) \Leftrightarrow a(W[s]) = 1,$
7. $W \models_{FTW(\mathbf{T})} OrderPoints(a, b) \Leftrightarrow$ *exist* $i, j \in \{1, ..., s\}$ *such that:*

$$i < j \ \wedge \ a(W[i]) = 1 \ \wedge \ b(W[j]) = 1.$$

It is worth noticing that the formulas of the language $FTW(\mathbf{T})$ may not only be satisfied by time windows from the set $TW(\mathbf{T})$ but also by time windows of the set $TW(\mathbf{T}')$ where $\mathbf{T}'$ is a temporal information system with an extended set of objects in relation to system $\mathbf{T}$.

Below we present several examples of formulas of the language FTW.

– If attribute a stores information about membership to the concept of *low speed*, then formula $EachPoint(a)$ describes a time window in which the vehicle's speed is low all the time.
– If attribute a stores information about membership to the concept of *accelerating*, then formula $ExistsPoint(a)$ describes time windows in which the vehicle happened to accelerate.
– If attribute a_1 stores information about membership to the concept of *accelerating* and attribute a_2 stores information about membership to the concept of *driving in the right lane*, then formula $ExistsPoint(a_1) \wedge EachPoint(a_2)$ describes the time window in which the vehicle happened to accelerate and the whole time drive in the right lane.

It is worthwhile mentioning that the language FTW defined above should be treated as an exemplary language for defining features of time windows, which has been used in experiments related to this paper. Obviously, it is possible to define many other languages of this type.

The FTW language formulas can be used to define patterns describing the properties of time windows, therefore, we call them *temporal patterns*.

Definition 27 (*A temporal pattern*). *Let* $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ *be a c-temporal information system. Any formula of the language* $FTW(\mathbf{T})$ *is called a temporal pattern of the system* $\mathbf{T}$.

Temporal patterns are often used in queries with binary answers such as *Yes* or *No*. For example, in the case of road traffic we have exemplary temporal patterns such as *Did vehicle speed increase in the time window?*, *Was the speed stable in the time window?*, *Did the speed increase before a move to the left lane occurred?* or *Did the speed increase before a speed decrease occurred?*.

We assume that any temporal pattern ought to be defined by a human expert using domain knowledge accumulated for the given complex dynamical system.

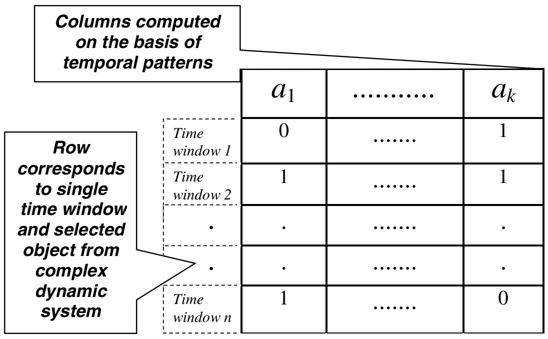

Columns computed on the basis of temporal patterns
Row corresponds to single time window and selected object from complex dynamic system
a_1
....
a_k
Time window 1
Time window 2
Time window n
0
1
1
1
1
0
....

Fig. 26. The scheme of an information system of time windows

6.7 Information System of Time Windows

The properties of the accessible time windows could be represented in a special information system which is called an information system of time windows (see also Fig. 26).

Definition 28 (*An information system of time windows*)**.** *Let us assume that:*

- $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ *is a c-temporal information system,*
- *s is a fixed length of time windows such that* $1 < s \leq card(U)$*,*
- $\Phi = \{\phi_1, ..., \phi_k\} \subseteq FTW(\mathbf{T})$ *is a family of temporal patterns defined by experts (sub-language of the language* $FTW(\mathbf{T})$*),*
- $\mathcal{P}_{FTW} = (\overline{U}, \Phi, \models_{FTW(\mathbf{T})})$ *is a property system, where* $\overline{U} = TW(\mathbf{T}, s)$*.*

The information system $\overline{\mathbf{T}} = (\overline{U}, \overline{A})$ *defined by the property system* $\mathcal{P}_{FTW}$ *is called an information system of time windows (TW-information system).*

Apparently, construction system $\overline{\mathbf{T}}$ requires generating the family of all time windows of established duration. Therefore, below we present the algorithm for generating all time windows of established duration (length) from a given c-system (see Algorithm 6.2).

On account of sorting objects in system $\mathbf{T}$, pessimistic time complexity of Algorithm 6.2 is of order $O(n \cdot log\ n)$, where n is the number of objects in the system $\mathbf{T}$.

Example 21. For the c-temporal information system from Example 19 an information system of time windows may be constructed. In order to do this, we may use for example temporal patterns chosen from the following collection of patterns:

1. low (moderate, high) vehicle speed at the first point (at the last point, at any point, at all points, at the minority of points, at the majority of points) of the time window,
2. low (moderate, high) maximal (minimal, average) speed of vehicle at the time window,

Algorithm 6.2. Generating all time windows

Input:
- temporal information system $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ such that $U = \{u_1, ..., u_n\}$,
- fixed length s of time windows such that $1 < s \leq n$.
- relation of linear order $\leq_{\{a_{id},a_t\}}$ defined on $U \times U$ in the following way:

$$\forall (u_1, u_2) \in U \times U : u_1 \leq_{\{a_{id},a_t\}} u_2 \Leftrightarrow u_1 \leq_{a_{id}} u_2 \wedge u_1 \leq_{a_t} u_2.$$

Output: The set of time windows $TW(\mathbf{T}, s)$

```
begin
    Create empty list $TW$ of windows
    Sort set $U$ using relation $\leq_{\{a_{id},a_t\}}$
    Create empty list $window$ of objects from the set $U$
    $currentID := a_{id}(u_1)$
    Insert $u_1$ to the list $window$
    for $i := 2$ to $n$ do
        if $(a_{id}(u_i) = currentID)$ then
            if $(Length(window) < s)$ then
                Add $u_i$ to the end of the list $window$
            else
                Remove the first object from the list $window$
                Add $u_i$ to the end of the list $window$
                Add $window$ to family $TW$
            end
        else
            Clear the list $window$
            $currentID := a_{id}(u_i)$
            Insert $u_i$ to the list $window$
        end
    end
    return $TW$
end
```

3. increasing (decreasing, maintaining) the speed of the vehicle (at the last point, at any point, at all points, at the minority of points, at the majority of points) of the time window,
4. low speed (moderate speed, high speed, increasing speed, decreasing speed, maintaining speed) in the time window before high speed (moderate speed, low speed, increasing speed, decreasing speed, maintaining speed) in the further part of the time window,
5. low (moderate, high) distance of the vehicle from the crossroads at the first point (at the last point, at any point, at all points, at the minority of points, at the majority of points) of the time window,

6. driving in the right (left) lane at the first point (at the last point, at any point, at all points, at the minority of points, at the majority of points) of the time window,
7. a slight turn towards the left (right) lane at the first point (at the last point, at any point, at all points, at the minority of points, at the majority of points) of the time window,
8. the location of the vehicle at the crossroads at the first point (at the last point, at any point, at all points, at the minority of points, at the majority of points) of the time window,
9. good (moderate, bad) visibility at the first point (at the last point, at any point, at all points, at the minority of points, at the majority of points) of the time window,
10. high humidity (low humidity, lack of humidity) of the road at the first point (at the last point, at any point, at all points, at the minority of points, at the majority of points) of the temporal.

It is easy to notice that each of the above patterns may be expressed in language FTW. □

The choice of specific patterns for the construction of the information system of time windows should depend on the concept which is to be approximated with the help of this system, obviously after performing the grouping (clustering) of objects (see Section 6.9).

6.8 Clustering Time Windows

The properties of time windows expressed by temporal patterns could be used for approximating temporal concepts which express more complex properties of time windows. However, it often happens that the objects of the information system of time windows (that is, system $\overline{\mathbf{T}}$) are not yet sufficient to use their properties for approximating temporal concepts. It is so due to the fact that there are too many of those objects and the descriptions of time windows which they represent are too detailed. Hence, if they are used for approximating temporal concepts, then the extension of the created classifier would be too small, which means that the classifier could classify too small number of tested objects.

Therefore, in this paper we use clustering (grouping) such objects which leads to obtaining a family of object clusters (clusters of time windows). From the general view point the grouping objects is always done using the language chosen by an expert and it is based on the fact that the clusters of objects are represented using formulas (patterns) defined in the language of grouping. Thanks to those patterns not only the objects of a given system are grouped but it is also possible to examine membership of other (tested) objects to individual clusters. Namely, we may say about the tested object that it belongs to the cluster when it satisfies the pattern describing this cluster.

In this paper, we propose the language $NL(\overline{\mathbf{T}})$ (neighborhood language) (see Definition 8) for grouping objects of system $\overline{\mathbf{T}} = (\overline{U}, \overline{A})$. The application of this language for object grouping requires defining the two following elements:

1. the dissimilarity function $DISM_{\overline{\mathbf{T}}}$ of object pairs in the information system $\overline{\mathbf{T}}$ (see Definition 7),
2. the family of formulas included in the set $NL(\overline{\mathbf{T}})$ which defines clusters of objects in system $\overline{\mathbf{T}}$ (see Definition 8).

We define the dissimilarity function using a dissimilarity classifier $\mu_{DISM_{\overline{\mathbf{T}}}}$ approximating it which is constructed for the dissimilarity table carefully specified by the expert.

However, defining the family of formulas included in $NL(\overline{\mathbf{T}})$, which defines clusters of objects in system $\overline{\mathbf{T}}$, requires the introduction of a subset of objects of system $\overline{\mathbf{T}}$ which are centers (generators) of clusters being created and the sequence of radiuses corresponding to them and limiting the clusters. Each atomic formula of the language $NL(\overline{\mathbf{T}})$ is, therefore, expression (u, ε) (where $u \in \overline{U}$ and $\varepsilon \in (0, 1]$), and at the same time such a formula encompasses all the objects for which the value of the dissimilarity function $DISM_{\overline{\mathbf{T}}}$ in relation to object u does not exceed value ε. Therefore, meanings of such formulas are in a sense neighborhoods of the objects membership to $\overline{U}$.

Algorithm 6.3. Clustering objects from an information system of time windows (version 1)

Input:
- information system of time windows $\overline{\mathbf{T}}=(\overline{U}, \overline{A})$ such that $\overline{U}=\{u_1, \ldots, u_n\}$,
- dissimilarity function $DISM_{\overline{\mathbf{T}}}$ of object pairs in the system $\overline{\mathbf{T}}$,
- deviation from standards ε.

Output: The family of formulas $F \subseteq NL(\overline{\mathbf{T}})$ defining clusters in the system $\overline{\mathbf{T}}$

```
1  begin
2      $F := \emptyset$
3      for $i := 1$ to $n$ do
4          Compute a neighborhood $N(u_i) = \{u \in \overline{U} : DISM_{\overline{\mathbf{T}}}(u_i, u) \leq \varepsilon\}$
5      end
6      Sort the set $\overline{U}$ in descending order // By sizes of neighborhoods
           computed in the previous step
7      repeat
8          Take object $u \in \overline{U}$ such that its neighborhood is maximal
9          $F := F \cup (u, \varepsilon)$
10         $\overline{U} := \overline{U} \setminus N(u)$
11     until $\overline{U} \neq \emptyset$
12     return $F$
13 end
```

The choice of the centers of those neighborhoods in order to construct formulas defining clusters should not be made at random. Objects being the centers of neighborhoods are called *standards* whereas the radiuses of neighborhoods *deviations* from the standards (see, *e.g.*, [89]). Both the standards and the deviations from them could be provided by experts. However, if this is for some reason difficult, there could be applied methods of determining standards and the deviations from them known from literature. It should be noted that in contemporary literature methods of this kind are often called *granulation methods* and the neighborhoods are called *granules* (see, *e.g.*, [89, 90, 91, 317, 319, 320, 321, 322, 323]).

In this paper, we propose the following clustering algorithm for automatic generation of object clusters, which is very similar to the algorithm presented in paper [165]. This algorithm is a greedy algorithm which initially chooses the object which has the biggest neighborhood and removes from the set being covered objects belonging to this neighborhood, thus choosing another neighborhood until it covers the whole set of objects (see Algorithm 6.3).

On the account of calculation of the neighborhoods for all the objects from set $\overline{U}$, the computational time complexity of Algorithm 6.3 is of order $O(n^2)$. In the case of bigger tables it may hinder or even make it impossible to effectively use this algorithm. Therefore, we also present a random version of the above algorithm (see Algorithm 6.4 and [165]).

As it can be observed, Algorithm 6.4 is in practice significantly faster in relation to Algorithm 6.3 because it does not require determining neighborhoods for all the objects from set $\overline{U}$, but only for the objects chosen randomly from set $\overline{U}$.

Algorithm 6.4. Clustering objects from an information system of time windows (version 2)

Input:
- information system of time windows $\overline{\mathbf{T}} = (\overline{U}, \overline{A})$ such that $\overline{U} = \{u_1, ..., u_n\}$,
- dissimilarity function $DISM_{\overline{\mathbf{T}}}$ of object pairs in the system $\overline{\mathbf{T}}$,
- deviation from standards ε.

Output: The family of formulas $F \subseteq NL(\overline{\mathbf{T}})$ defining clusters in the system $\overline{\mathbf{T}}$

```
1  begin
2      $F := \emptyset$
3      repeat
4          Randomly select $u \in \overline{U}$
5          Compute a neighborhood $N(u) = \{v \in \overline{U} : DIST_{\overline{\mathbf{T}}}(u, v) \leq \varepsilon\}$
6          $F := F \cup (u, \varepsilon)$
7          $\overline{U} := \overline{U} \setminus N(u)$
8      until $\overline{U} \neq \emptyset$
9      return $F$
10 end
```

From the formal viewpoint, clusters of time windows are defined using the language *ECTW*.

Definition 29 (*A language for extracting clusters of time windows*)**.** *Let us assume that:*

– $\mathbf{T} = (U,\, A,\, a_{id},\, \leq_{a_{id}},\, a_t,\, \leq_{a_t})$ *is a c-temporal information system,*
– $\overline{\mathbf{T}} = (\overline{U}, \overline{A})$ *is information system of time windows for the system* $\mathbf{T}$.

Any neighborhood language $NL(\overline{\mathbf{T}})$ *is called a language for extracting clusters of time windows from system* $\mathbf{T}$ *(denoted by* $ECTW(\mathbf{T})$ *or ECTW-language, when* $\mathbf{T}$ *is fixed).*

The formulas of the language *ECTW* describe clusters of time windows from the temporal information system.

Definition 30 (*A cluster of time windows*)**.** *Let* $\mathbf{T} = (U,\, A,\, a_{id},\, \leq_{a_{id}},\, a_t,\, \leq_{a_t})$ *be a c-temporal information system. The meaning any formula* $\phi \in ECTW(\mathbf{T})$ *is called a cluster of time windows from the system* $\mathbf{T}$.

In the example below we illustrate how the dissimilarity function may be defined for the time windows grouping.

Example 22. For the information system of time windows obtained in Example 21, a dissimilarity function may be constructed on the basis of expert knowledge. In order to do this, the value of the dissimilarity function for a certain (the most representative set of object pairs of this system) should be obtained from the experts. To make it happen, the expert should know what temporal concept is approximated using clusters obtained with the help of currently defined dissimilarity function (see Section 6.9). Let us assume that it is the concept *accelerating in the right lane.* In terms of this concept the expert may immediately define several vehicle groups which behave very similarly, that is, vehicles being in one of such groups, in terms of the dissimilarity function being constructed, should not differ too much. For example, they may be the following vehicle groups:

A. vehicles which accelerate in a given time window but they do it with a smaller or bigger intensity,
B. vehicles which accelerate in a given time window, however, they do not do it in a continuous manner, sometimes maintaining stable speed but they never decrease it,
C. vehicles which in a given time window increase, decrease or at times maintain stable speed,
D. vehicles which decrease speed in a given time window but they do it with a smaller or bigger intensity.

In Table 4, values of chosen attributes (temporal patterns) are presented for objects u_1, u_2, u_3, u_4, and u_5 of a certain information system of time windows.

Table 4. Exemplary objects of some TW-information system

No	**Temporal patterns**	u_1	u_2	u_3	u_4	u_5
1.	High speed of the vehicle at the first point of the time window	0	0	0	0	1
2.	Moderate speed of the vehicle at the first point of the time window	0	0	0	1	0
3.	Low speed of the vehicle at the first point of the time window	1	1	1	0	0
4.	Increasing the speed of the vehicle at all points of the time window	1	1	0	0	0
5.	Increasing the speed of the vehicle at some point of the time window	1	1	1	1	0
6.	Maintaining stable speed of the vehicle at all points of the time window	0	0	0	0	0
7.	Maintaining stable speed of the vehicle at some time window point	0	0	1	1	0
8.	Decreasing speed of the vehicle at all points of the time window	0	0	0	0	1
9.	Decreasing speed of the vehicle at some point of the time window	0	0	0	1	1
10.	High speed of the vehicle at the last point of the time window	0	1	0	1	0
11.	Moderate speed of the vehicle at the last point of the time window	1	0	1	0	0
12.	Low speed of the vehicle at the last point of the time window	0	0	0	0	1

It is easy to notice that object u_1 belongs to the group of vehicles A and does not belong to the groups B, C and D. Similarly, object u_2, which is presented in the table, also belongs to group A. The reason for this is the fact that both objects increased the speed the whole time, but object u_2 at the end of the time window reached a high speed and object u_1 only moderate speed. That is why, the dissimilarity function value for these two objects is low, that is, 0.1, whereas object u_3 belongs to group B, for it maintained stable speed throughout a part of the window. Therefore, the dissimilarity function between objects u_2 and u_3 is bigger and equals 0.25. Object u_4 belongs to group C and therefore the difference between this object and object u_1 is 0.5. Finally, the dissimilarity function value between objects u_2 and u_5 (u_5 belongs to group D) is the biggest and is as big as 1.0, for in terms of the concept *accelerating in the right lane* these objects differ to the maximum because object u_1 is increasing speed and object u_5 is decreasing speed all the time. The dissimilarity function values proposed for all object pairs are compared in Table 5. Finally, we notice that objects u_1, u_3, u_4 and u_5 may be treated as standards for which the deviation is 0.1. $\square$

Table 5. Values of dissimilarity function for pairs of time windows

	u_1	u_2	u_3	u_4	u_5
u_1	0	0.1	0.25	0.5	1.0
u_2	0.1	0	0.25	0.5	1.0
u_3	0.25	0.25	0	0.25	0.75
u_4	0.5	0.5	0.25	0	0.5
u_5	1.0	1.0	0.75	0.5	0

If there is a family of standards given along with their deviations, then on their basis we can construct a family of formulas of the language $NL(\overline{\mathbf{T}})$ which represent the established clusters in system $\overline{\mathbf{T}}$. Now, in order to construct a new information system which represents the clusters' properties we need to define the language defining the cluster features.

Definition 31 (*A language for defining features of clusters of time windows*). *Let* $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ *be a c-temporal information system and* $\overline{\mathbf{T}} = (\overline{U}, \overline{A})$ *be an information system of time windows for the system* $\mathbf{T}$. *A language for defining features of clusters of time windows for the system* $\mathbf{T}$ *(denoted by* $FCTW(\overline{\mathbf{T}})$ *or* $FCTW$*-language, when* $\overline{\mathbf{T}}$ *is fixed) is defined in the following way:*

- *the set* $AL_{FCTW}(\mathbf{T}) = AL_{FTW}(\mathbf{T}) \cup \{ExistsWindow, EachWindow, MajorityWindows, MinorityWindows\}$ *is an alphabet of the language* $FCTW(\overline{\mathbf{T}})$,
- *for any* $\alpha, \beta \in FCTW(\overline{\mathbf{T}})$ *expressions of the form:* $ExistsWindow(\alpha)$, $EachWindow(\alpha)$, $MajorityWindows(\alpha)$, $MinorityWindows(\alpha)$ *are atomic formulas of the language* $FCTW(\overline{\mathbf{T}})$.

Now, we define the semantics of the language $FCTW(\overline{\mathbf{T}})$. The formulas of the language $FCTW(\overline{\mathbf{T}})$ may be treated as the descriptions of families of time windows in system $\mathbf{T}$. For example, formula $ExistsWindow(\alpha)$ is interpreted as the description of all those families of time windows of system $\mathbf{T}$ in which there exists at least one such window that satisfies formula α.

Definition 32. *Let us assume that:*

– $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ *is a c-temporal information system,*
– $\overline{\mathbf{T}} = (\overline{U}, \overline{A})$ *is a information system of time windows for the system* $\mathbf{T}$.

The satisfiability of an atomic formula $\alpha \in FCTW(\overline{\mathbf{T}})$ *by a family of time windows* $\mathcal{W} = \{W_1, ..., W_k\} \subseteq TW(\mathbf{T})$ *(denoted by* $\mathcal{W} \models_{FCTW(\mathbf{T})} \alpha$*), is defined in the following way:*

1. $\mathcal{W} \models_{FCTW(\mathbf{T})} ExistsWindow(\alpha) \Leftrightarrow \exists_{W \in \mathcal{W}}\ W \models_{FTW(\mathbf{T})} \alpha$,
2. $\mathcal{W} \models_{FCTW(\mathbf{T})} EachWindow(\alpha)) \Leftrightarrow \forall_{W \in \mathcal{W}}\ W \models_{FTW(\mathbf{T})} \alpha$,
3. $\mathcal{W} \models_{FCTW(\mathbf{T})} MajorityWindows(\alpha)) \Leftrightarrow$

$$card(\{W \in \mathcal{W} : W \models_{FTW(\mathbf{T})} \alpha\}) > \lfloor (k-1)/2 \rfloor,$$

4. $\mathcal{W} \models_{FCTW(\mathbf{T})} MinorityWindows(\alpha)) \Leftrightarrow$

$$card(\{W \in \mathcal{W} : W \models_{FTW(\mathbf{T})} \alpha\}) < \lceil (k-1)/2 \rceil.$$

Below we present several examples of formulas of the language $FCTW$.

- If attribute a_1 of system $\mathbf{T}$ stores information about membership to the concept of *accelerating*, then formula $ExistsWindow(LastPoint(a_1))$ describes such clusters of time windows where there is a window in which acceleration occurred at the last point of this window.
- If attribute a_1 of system $\mathbf{T}$ stores information about membership to the concept of *accelerating*, then formula $MajorityWindows(EachPoint(a_1))$ describes such clusters of time windows that in the majority of them the speed is being increased all the time.
- If attributes a_1 and a_2 of system $\mathbf{T}$ store information about membership respectively to the concepts of *accelerating* and *decelerating*, then formula $\neg ExistsWindow(ExistsPoint(a_2)) \wedge EachWindow(ExistsPoint(a_1))$ describes such clusters of time windows in which there does not exist a single window in which deceleration occurred and in all the windows of this cluster the speed is increased.

The patterns of the language $FCTW$ could be utilized to define the properties of clusters of time windows. Due to this clusters of time windows are represented by information systems which we call an information system of clusters of time windows.

Definition 33 (*An information system of clusters of time windows*)**.** *Let us assume that:*

- $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ *is a c-temporal information system,*
- $\overline{\mathbf{T}} = (\overline{U}, \overline{A})$ *is a information system of time windows for the system* $\mathbf{T}$*.*
- $\psi_1, ..., \psi_n \in ECTW(\mathbf{T})$ *is a defined by experts family of temporal patterns corresponding to clusters of time windows* $CL_1, ..., CL_n$*,*
- $\Phi = \{\phi_1, ..., \phi_m\} \in FCTW(\mathbf{T})$ *is a defined by experts family of features of clusters of time windows,*
- $\mathcal{P}_{FCTW} = (\overline{\overline{U}}, \Phi, \models_{FCTW(\mathbf{T})})$ *is a property system, where* $\overline{\overline{U}} = \{CL_1, ..., CL_n\}$*.*

The information system $\overline{\overline{\mathbf{T}}} = (\overline{\overline{U}}, \overline{\overline{A}})$ *defined be the property system* $\mathcal{P}_{FCTW}$ *is called an information system of clusters of time windows (CTW-information system).*

It is easy to see that construction of the system $\overline{\overline{\mathbf{T}}}$ requires generating clusters of time windows in such a way that it could be possible to check the ability to satisfy formulas of the language $FCTW(\overline{\mathbf{T}})$. Such clusters may be generated by the linear overview of set $\overline{U}$ and assigning the objects of this set to individual clusters. The complexity of such an algorithm would be of order $O(l \cdot n)$, where n is the number of clusters and $l = card(\overline{U})$.

Example 23. Using the dissimilarity function from Example 22, the objects of the information system of time windows from Example 21 may be clustered. However, in order to construct an information system for the determined family of clusters, patterns should be established. They may be the following:

1. in each time window of a cluster there occurs speed increase the whole time (the pattern describes the properties of vehicles from group A from Example 22),
2. in each time window of a cluster there occurs speed increase (the pattern describes the properties of vehicles from groups A, B, C from Example 22),
3. in the majority of time windows of the cluster there does not occur speed increase,
4. in the cluster there exists a time window in which there occurs speed decrease (the pattern describes the properties of vehicles from groups C and D from Example 22),
5. in each time window of the cluster at the first time point of the window the speed is low and at the last point of the window it is high or moderate,
6. in most time windows of the cluster the vehicles drive in the right lane the whole time. □

The choice of specific patterns to construct the information system of clusters of time windows depends on the temporal concept which is approximated with the help of this system (see Section 6.9).

6.9 Temporal Concept Table

The properties of clusters of time windows expressed by formulas of the language *FCTW* may be used for constructing decision tables which enable the identification of temporal concepts. For this purpose, it is necessary to add to system $\overline{\overline{\mathbf{T}}}$ a decision attribute which characterizes the cluster's membership to the established temporal concept. In this way, we obtain a decision table which is called the temporal concept table (see Fig. 27).

Definition 34 (*A temporal concept table*). *Let us assume that:*

– $\overline{\overline{\mathbf{T}}} = (\overline{\overline{U}}, \overline{\overline{A}})$ *is a information system of clusters of time windows,*
– C *is concept defined in the set* $\overline{\overline{U}}$.

A temporal concept table for the concept C *is a decision table* $\overline{\overline{\mathbf{T}}}_C = (\overline{\overline{U}}, \overline{\overline{A}} \cup \{d_C\})$, *where attribute* d_C *is a membership function of the concept* C.

A question arises, how it is possible that an expert can propose the value of decision attribute d_C. To make a decision concerning the value of this attribute an expert has at his disposal the values of attributes from set $\overline{\overline{A}}$. These are the attributes proposed earlier as the characteristic features of clusters of time windows. The expert could therefore propose them in such a way that he or she is now able to use them successively in order to determine the membership of

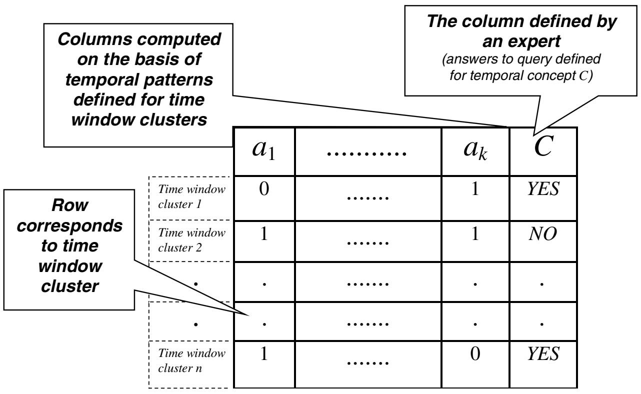

Columns computed on the basis of temporal patterns defined for time window clusters
The column defined by an expert (answers to query defined for temporal concept C)
Row corresponds to time window cluster
a_1 .... a_k C
Time window cluster 1 0 .... 1 YES
Time window cluster 2 1 .... 1 NO
. . ..
. . ..
Time window cluster n 1 .... 0 YES

Fig. 27. The scheme of a temporal concept table

the whole clusters to concept C. In other words the properties of clusters have to be defined in such a way that they could serve to determine the time window clusters' membership to temporal concept C.

Example 24. The information system obtained on the basis of patterns such as in Example 23 may be enriched by a decision attribute which describes the membership of individual clusters of time windows to the established temporal concept (*e.g.*, the concept of *accelerating in the right lane*). In this way we obtain a decision table which may serve to the approximation of this concept. □

If there is given set H of linearly ordered layer labels of concept C, then for table $\overline{\overline{\mathbf{T}}}_C$ there could be constructed a stratifying classifier μ_C^H which can indicate, for each time window cluster, the layer of concept C to which this cluster belongs.

6.10 Classification of Time Windows

In practical applications we encounter a question whether the behavior of the complex objects observed in the time window belongs to a given temporal concept defined for clusters of time windows. It means the classification of time windows to the concept determined on the set of clusters of time windows. Meanwhile, the classifier constructed for table $\overline{\overline{\mathbf{T}}}_C$ can classify window clusters and not single time windows. Therefore, before we use such a classifier, it should be checked to which cluster of time windows the tested time window belongs. That is how the algorithm of time window classification works (see Algorithm 6.5). However, we assume that during execution of the Algorithm 6.5 the following elements are available (established or computed earlier):

– a training c-temporal information system $\mathbf{T}$,
– a fixed length l of time windows,
– a family of temporal patterns $\phi_1, ..., \phi_m \in FTW(\mathbf{T})$,
– a system $\overline{\mathbf{T}} = (\overline{U}, \overline{A})$ such that attributes from the set $\overline{A}$ correspond to formulas $\phi_1, ..., \phi_m$,

- a dissimilarity function $DISM_{\overline{\mathbf{T}}}$ of object pairs from the system $\overline{\mathbf{T}}$, approximated using the stratifying classifier $\mu_{DISM_{\overline{\mathbf{T}}}}$,
- a family of temporal patterns $\psi_1, ..., \psi_n \in ECTW(\mathbf{T})$ such that $\psi_i = (std_i, \varepsilon_i)$, for $i = 1, ..., n$,
- a system $\overline{\overline{\mathbf{T}}} = (\overline{\overline{U}}, \overline{\overline{A}})$ such that objects from the set $\overline{\overline{U}}$ correspond to clusters defined by formulas $\psi_1, ..., \psi_n$,
- a concept C defined in the set $\overline{\overline{U}}$,
- a decision table $\overline{\overline{\mathbf{T}}}_C = (\overline{\overline{U}}, \overline{\overline{A}} \cup \{d_C\})$ with decision attribute representing membership of objects from the set $\overline{\overline{U}}$ to the concept C,
- a linearly ordered set $(H, \leq_H)$ such that H is a set of layer labels of the concept C and $\leq_H$ is a relation of linear order on the set H,
- a stratifying classifier μ_C^H constructed for the concept C on the basis the table $\overline{\overline{\mathbf{T}}}_C$,
- a test c-temporal information system $\mathbf{T}_{TS}$.

Because the Algorithm 6.5 classifies time windows to one layer of concept C, it is the classifier stratifying temporal concept C.

Assuming that all operations of the examination of formulas satisfiability, the computation of the dissimilarity function values and application of the classifier μ_C^H (occurring in the above algorithm) are executed at the constant time, then

Algorithm 6.5. Time window classification (*ClassifyWindow*)

```
Input: A test time window $W \in TW(\mathbf{T}_{TS}, l)$
Output: The layer of concept $C$

1  Procedure $ClassifyWindow(W)$
2  begin
3      Create empty list row
4      for $\phi_1, ..., \phi_m$ do
5          if $W \models_{FTW(\mathbf{T})} \phi_i$ then
6              Add '1' to the end of the list row
7          else
8              Add '0' to the end of the list row
9          end
10     end
11     Create new object $u_{row}$ of the system $\overline{\mathbf{T}}$ on the basis values of
       attributes from the list row.
12     Select a formula $\psi = (std, \varepsilon) \in \{\psi_1, ..., \psi_n)$ such that:
13       1. $u_{row} \models_{ECTW(\mathbf{T})} \psi$ and
14       2. $DISM_{\overline{\mathbf{T}}}(u_{row}, std) = min_{i \in \{1,...,n\}}\{DISM_{\overline{\mathbf{T}}}(u_{row}, std_i)\}$
15     Select object $u_\psi \in \overline{\overline{U}}$ corresponding to the formula $\psi$
16     return $\mu_C^H(u_\psi)$
17 end
```

the time complexity of the above algorithm is of order $O(m + n)$, where m is the number of temporal patterns for time windows used and n is the number of time window clusters.

6.11 Behavioral Graphs

Temporal concepts defined for objects from a complex dynamical system can be treated as nodes of a graph called a *behavioral graph*, where connections between nodes represent temporal dependencies. Such connections between nodes can be defined by an expert or read from a data set that has been accumulated for a given complex dynamical system.

Definition 35 (*A behavioral graph*)**.**

1. *A behavioral graph G is an ordered pair (V, E), where V is a nonempty and finite set of nodes (temporal concepts) and E is a set of directed edges $E \subset V \times V$ (connections represent temporal dependencies between temporal concepts).*
2. *A temporal path in the behavioral graph $G = (V, E)$ is a sequence of nodes $v_1, ..., v_l$ such that for any $i \in \{1, ..., l-1\}$ an edge $(v_i, v_{i+1}) \in E$. A number l is called a length of temporal path $v_1, ..., v_l$.*
3. *The family of all temporal paths with the length l $(l > 0)$ in the behavioral graph G is denoted by $PATH(G, l)$.*

Bellow, we present an example of behavioral graph.

Example 25. Fig. 28 presents an example of behavioral graph for a single object-vehicle exhibiting a behavioral pattern of vehicle while driving on a road. In this behavioral graph, for example, connections between node *Acceleration on the right lane* and node *Acceleration and changing lanes lanes from right to left* indicates that after an acceleration in the right lane, a vehicle can change to the left lane (maintaining its acceleration during both time windows). □

In addition, a behavioral graph can be constructed for different kinds of objects such as single vehicles or groups of vehicles and defined for behaviors such as driving on the strength road, driving through crossroads, overtaking, and passing. Therefore, we consider any behavioral graph as a model for behavioral patterns (see Section 6.23).

6.12 Representing Spatial Properties of Structured Objects Using Concepts

If we wish to observe the behavior of structured objects changing over time, then it turns out that observing the behaviors of individual parts of these objects separately is not sufficient. It happens that way because if we observe the behavior of a certain structured object we have to consider the issue what relations there are between the types of behaviors of individual parts of this object and how these relations coexist and change over time. For example, if we observe the

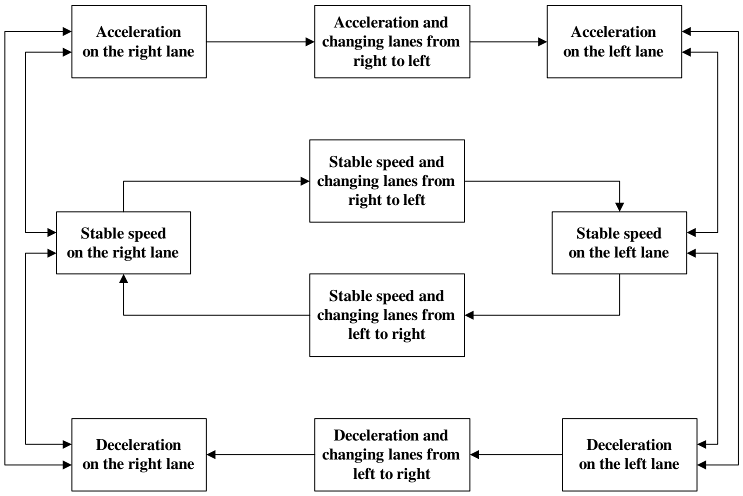

Acceleration on the right lane
Acceleration and changing lanes from right to left
Acceleration on the left lane
Stable speed and changing lanes from right to left
Stable speed on the right lane
Stable speed on the left lane
Stable speed and changing lanes from left to right
Deceleration on the right lane
Deceleration and changing lanes from left to right
Deceleration on the left lane

Fig. 28. A behavioral graph for a single object-vehicle

overtaking maneuver made by the structured object, which is a pair of vehicles (the overtaking and overtaken vehicles), then we are interested in the relations between the behavior of the overtaking vehicle and the behavior of the overtaken vehicle. Another example may concern the observation of the courses of illnesses, which being in the interaction with one another, develop in a given patient.

The spatial properties of such bounds may be represented using concepts which concern the sets of structured object parts. Such concepts represent the partition of all structured objects into those which belong to the concept and those structured objects which do not belong to the concept being concerned. Examples of such concepts may be concepts concerning the distance between two chosen vehicles belonging to the group of vehicles being examined (*e.g.*, short distance between two vehicles, long distance between two vehicles, driving on the same lane).

Similarly to the concepts representing spatial properties of unstructured objects, the concepts representing spatial properties of structured objects also may be approximated. The approximation may take place on the basis of appropriately constructed for this purpose decision table. Each object of such a table corresponds to a certain structured object. Conditional attributes are the arrangement of attributes from a given temporal information system registering the parameters of individual parts of a given structured object. However, the decision attribute of this table describes the membership of objects of the table under construction to the approximated concept concerning the spatial property of the whole structured object and its values are suggested by the expert on the basis of domain knowledge. The classifier constructed for such a table allows testing any structured object for the membership to the approximated concept. Due to the fact that the approximated concept is spatial (it does not require following changes over time with the exception of parameter changes since the last observation of the structured object), we can apply here classical classifiers, stratifying classifiers as well as classifiers based on AR-schemes. However, during the construction of the decision table for the purpose of approximation of spatial concepts for structured objects we encounter a serious problem concerning extraction of relevant context for the parts of structured objects. One of the solutions to this problem could be application of *a sweeping algorithm around the complex object* which is characterized in the following subsection.

6.13 Sweeping Algorithm around the Complex Object

In the paper, each structured object occurring in a complex dynamical system is understood as an object consisting of parts, that is, objects of lesser complexity which are linked with one another by relations describing the context in which individual parts of a structured object occur. Therefore, both learning concepts concerning structured objects and testing structured objects for membership to such concepts requires a method of isolating structured objects under examination. Unfortunately, the elementary approach to isolation of structured objects which consists in analyzing all the subsets (of established number) from the existing set of potentially parts of structured objects cannot be used because of the high computational complexity of algorithms generating and examining such subsets. For example, if we examine a system which has 20 complex objects (*e.g.*, 20 vehicles on the road) and we are interested in structured objects defined as groups of 6 objects (6 vehicles as in the case of examining the dangerous overtaking maneuver) (see Fig. 55) then the general number of groups which require observation is equal to the number of 6-element combinations from the 20-element set, that is, $\binom{20}{6} = 38760$. Additionally, if the observation is carried out only over 100 time units, then the temporal information system describing the properties of all such groups of vehicles would have to have almost 4 million rows!

Another possibility is the application of methods which use the context in which the objects being parts of structured objects occur. This type of methods isolates structured objects not by direct indication of the set of parts of the searched structured object but by establishing one part of the searched structured object and attaching to it other parts, being in the context to the established part. Unfortunately, also here, the elementary approach to determining the context of the part of the structured object, consisting in examining all possible subsets (of established number) of the set of potential structured objects to which the established part of the structured object belongs, cannot be applied because of a great number of such subsets. For example, in order to identify a group of vehicles which are involved in a dangerous maneuver and to which the established vehicle belongs (to which we pay attention), it would be necessary to follow (in real time) the behavior of all possible groups of vehicles of the established number (*e.g.*, six vehicles) to which the established vehicle belongs, which is, with a relatively small number of visible vehicles, still too difficult.

Hence, we need special methods of determining the context of the established part of the structured object based on domain knowledge, which allows to limit the number of analyzed sets of parts of structured objects. Therefore, in this paper we propose a special method which we call *the sweeping method*. The method works in the following way: at the stage of learning the behavior for structured objects only those structured objects are taken into account that came into being through the so-called sweeping around of all the complex objects which may be part of a structured object, and at the same time for each such object there is only one structured object constructed. The activity of sweeping is carried out by the special sweeping algorithm which returns to the established complex object an ordered list of these objects which are significant from the point of view of the type of examined complex behavior of structured objects. The organization of this list is important due to the fact that in the obtained structured object represented by the list, each object (a part of a structured object) can have its role in the examined complex behavior of structured objects.

Definition 36 (*A sweeping algorithm around the complex object*)**.** *Let* $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ *be a temporal information system such that objects from the set* U *are unstructured objects of a fixed type* T*. Each algorithm* SA *which at the input receives a chosen object* $u \in U$*, and at the output returns the subset of* U *such that the object* u *belongs to this subset represented in the form of organized* k*-element list* $SA(u)$ *(where* $k > 1$*) is called a sweeping algorithm around the complex object.*

The sweeping algorithms must be constructed individually on the basis of domain knowledge for each complex behavior which is to be identified. By this we mean such complex types of behavior as: overtaking vehicles on the road, driving of a group of vehicles in a traffic jam, chasing one car after another, persistence of respiratory insufficiency in patients. Also, the parameter k should be fixed individually for a given sweeping algorithm.

As a result of applying the sweeping algorithm we only obtain as many structured objects as many complex objects constituting potential parts of structured objects there are, for each of the established structured objects gets attached to one unstructured object (one of its parts) which plays a specific role in the structured object. For example, if we examine the behavior of a group of vehicles connected with the overtaking maneuver, then the vehicle distinguished during the sweeping algorithm's activity may be the vehicle which overtakes. In this group also other vehicles may be distinguished (the overtaken vehicle, the oncoming vehicle, the vehicle driving in the back, etc.) which takes place when we apply other sweeping algorithms (constructed for this type of objects).

We present an example of the sweeping algorithm for the purpose of recognizing the overtaking maneuver, where the distinguished overtaking vehicle is the complex object (see Algorithm 6.6). This algorithm regards the situation presented in the Fig. 29. The group of vehicles which are returned by Algorithm 6.6 which is the so called *sweeping zone* comprises 6 vehicles which are the most important from the point of view of planning and performing the overtaking maneuver by a given vehicle (see Appendix A).

Let us notice that in the Algorithm 6.6 a number of auxiliary concepts are used. They are for example such concepts as: *going close to u*, *going in the right lane*, *going in front of vehicle u*, *going behind vehicle u*, *going in the same*

Algorithm 6.6. The sweeping algorithm for the purpose of recognizing the overtaking maneuver

```
Input:
  – temporal information system $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ such that
    any object $u \in U$ is an object of the fixed type $T$ (a single vehicle),
  – object $u \in U$

Output: The sweeping zone around $u$
1  begin
2      Create empty list $L$.
3      Insert object (vehicle) $u$ to the list $L$.
4      If there is a vehicle in the right lane close behind to vehicle $u$ and going
       in the same direction, add it to list $L$ as vehicle BR (see Fig. 29).
5      If there is a vehicle in the left lane close behind to vehicle $u$, going in
       the same direction, add it to list $L$ as vehicle BL (see Fig. 29).
6      If there is a vehicle close in front of vehicle $u$ in the right lane, going
       in the same direction, add it to list $L$ as vehicle FR1 (see Fig. 29).
7      If during the previous stage you added a vehicle to list $L$ as vehicle
       $FR1$, then if in front of this vehicle in the right lane there is another
       vehicle going close in the same direction as vehicle $u$, then add it to
       list $L$ as vehicle FR2 (see Fig. 29).
8      If there is an oncoming vehicle in front of vehicle $u$ in the left lane,
       then add it to list $L$ as vehicle FL (see Fig. 29).
9      return $L$ as $SA(u)$.
10 end
```

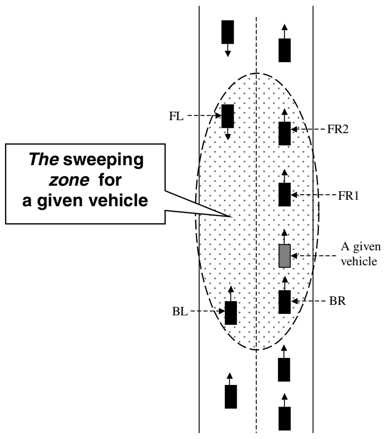


Fig. 29. A given vehicle and its sweeping zone

direction as u, *oncoming vehicle*. All these concepts require approximation, but they are spatial concepts and therefore they are much easier to be approximated than the spatio-temporal concepts. It also means that in the above algorithm the application of classifiers approximating these concepts is necessary.

One should, however, be aware of the fact that each sweeping algorithm works on the basis of the sweeping heuristics defined by an expert. Therefore, such an algorithm isolates only structured objects suggested by this heuristics. That is why, it is only a very selective perception and its application requires a great support from the experts who have to provide all crucial sweeping heuristics, in terms of perception of the whole complex dynamical system.

During experiments with the road simulator (see Section 6.25) the sweeping algorithm already worked at the stage of generating data using the simulator. Therefore, in the data set there is already available information about structured objects consisting of 6 vehicles connected with the identification of the overtaking maneuver. Whereas, in experiments with medical data (see Section 6.26), where a group of illnesses is the structured object, the performance of the sweeping algorithm is based on constructing only such groups of illnesses which occurred in particular patients from the data at the same time.

6.14 C-temporal Information System for Structured Objects

The sweeping algorithm, defined in Section 6.13, efficiently extracts structured objects. For objects extracted in such a way we may define concepts which may be approximated using attributes of a given temporal information system. Thanks to that c-temporal information systems may be constructed whose attributes describe spatial relation properties between parts of structured objects. However, because the attributes of such c-system are to concern relation properties between structured objects extracted with the sweeping algorithm, it comes into being in a slightly different way than the standard c-system, that is, a special algorithm is needed which constructs c-system on the basis of the available temporal information system with the use of the sweeping algorithm. Such an algorithm we call a *cr-transformation.*

Definition 37. *Let us assume that:*

- $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ *is a c-temporal information system such that any object* $u \in U$ *is an object of the fixed type* T*,*
- $C_1, ..., C_k$ *is a family of spatial concepts determined for structured objects (parts of such objects must be registered at the same time),*
- $\mu_1, ..., \mu_k$ *is a family of classifiers which approximate concepts* $C_1, ..., C_k$ *on the basis of the parameters of the structured object parts established on the basis of attributes from set* $A \setminus \{a_{id}, a_t\}$*,*
- SA *is a sweeping algorithm around objects from the system* $\mathbf{T}$*.*

1. *An operation of changing the system* $\mathbf{T}$ *to the c-system*

$$\mathbf{T}_r = (U_r, A_r, c_{id}, \leq_{c_{id}}, c_t, \leq_{c_t})$$

is called a cr-transformation of the system $\mathbf{T}$ *iff*

- *$U_r = \{g_1, ..., g_n\}$, where $g_i = SA(u_i)$ is the result returned by the sweeping algorithm SA for u_i, for $i = 1, ..., n$,*
- *$A_r = \{c_{id}, c_t, c_1, ..., c_k\}$ where attributes from the set A_r are defined in the following way:*
  - *$\forall g_i \in U_r : c_{id}(g_i) = a_{id}(u_i) \ \wedge \ c_t(g_i) = a_t(u_i)$,*
  - *attributes $c_1, ..., c_k$ represent concepts $C_1, ..., C_k$ where:*

$$\forall_{g \in U_r} \ c_i(g) = \mu_i(g) \text{ for } i = 1, ..., k.$$

2. The c-system $\mathbf{T}_r$ is called a result of cr-transformation of the system $\mathbf{T}$.

We present the algorithm of the cr-transformation for the temporal information system (see Algorithm 6.7).

Algorithm 6.7. Cr-transformation

```
Input:
  1. $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ is a c-temporal information system such
     that any $u \in U$ is a concept object of the fixed type $T$,
  2. $C_1, ..., C_k$ is a family of spatial concepts determined for structured
     objects (parts of such objects must be registered at the same time),
  3. $\mu_1, ..., \mu_k$ is a family of classifiers which approximate concepts $C_1$,
     $..., C_k$ on the basis of the parameters of the structured object parts
     established on the basis of attributes from set $A \setminus \{a_{id}, a_t\}$,
  4. $SA$ is a sweeping algorithm around objects from the system $\mathbf{T}$.

Output: The c-information temporal system
        $\mathbf{T}_r = (U_r, A_r, c_{id}, \leq_{c_{id}}, c_t, \leq_{c_t})$
begin
    Create an empty information system $\mathbf{T}_r$ which has attributes
        $c_{id}, c_t, c_1, ..., c_k$ where attributes $c_{id}$ and $c_t$ are of the identical type
        as their counterparts in system $\mathbf{T}$ and attributes $c_1, ..., c_k$ are
        binary attributes // $\mathbf{T}_r$ is without any objects for the
        time being
    for $i := 1$ to $card(U)$ do
        Compute a structured object $g := SA(u_i)$.
        Create an empty list of values $L$.
        Add $a_{id}(u_i)$ to the list $L$.
        Add $a_t(u_i)$ to the list $L$.
        for $j = 1$ to $k$ do
            Add $\mu_j(g)$ to the list $L$.
        end
        Add new object represented by values from $L$ to the system $\mathbf{T}_r$.
    end
    return $\mathbf{T}_r$
end
```

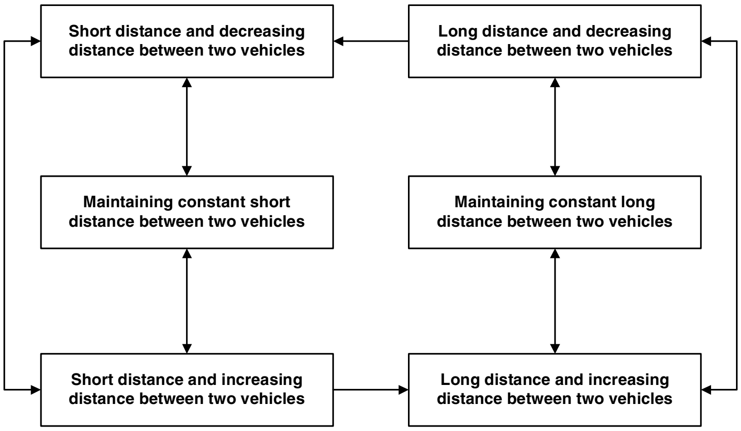

Short distance and decreasing distance between two vehicles
Long distance and decreasing distance between two vehicles
Maintaining constant short distance between two vehicles
Maintaining constant long distance between two vehicles
Short distance and increasing distance between two vehicles
Long distance and increasing distance between two vehicles

Fig. 30. A behavioral graph for relations between two vehicles exhibiting a changes of distance between vehicles while driving on a road

Assuming that each classifier $\mu_1, ..., \mu_k$ and algorithm SA work over the time of order $O(C)$, where C is a certain constant, then the time complexity of the Algorithm 6.7 is of order $O(n \cdot k)$, where $n = card(U)$ and k is the number of concepts used to construct attributes.

The result of the performance of the algorithm of the cr-transformation is the c-system and therefore it may be used to approximate temporal concepts describing the relation between the structured object's parts. Mechanisms of performing such approximation are the same as in the case of temporal concepts concerning unstructured complex objects. Finally, it leads to the behavioral graph describing complex objects being parts of structured objects with connection to the types of behavior of other parts of these structured objects.

Fig. 30 presents an example of behavioral graph for relations between two objects (vehicles) exhibiting a changes of distance between vehicles while driving on a road.

6.15 Sequences of Time Windows

Complex types of behavior of complex objects, treated as unstructured objects may be described using behavioral graphs of complex objects. As a result of temporal concept approximation classifiers may be obtained for all concepts occurring in behavioral graphs which enable the examination of objects' membership to the temporal concepts. Hence, behavioral graphs and classifiers for temporal concepts allow to follow the types of behavior of the complex objects over a longer period of time than the time window. This longer period we call *a sequence of time windows*. Therefore, the learning of the perception of the complex behavior of structured objects with the use of the gathered data and behavioral graphs constructed for all parts of structured objects of a given type, as well as the further use of the learned classifiers to identify the types of behavior of structured objects, requires working out the mechanisms of the extraction of sequences of time windows. Therefore, we need a language for extraction of sequences of time windows.

Definition 38 (*A language for extracting sequences of time windows*). *Let* $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ *be a c-temporal information system and let* $\mathbb{Z}_1$ *be the set of integer numbers equal or greater than 1. A language for extracting sequences of time windows from system* $\mathbf{T}$ *(denoted by* $ESTW(\mathbf{T})$ *or* $ESTW$*-language, when* $\mathbf{T}$ *is fixed) is defined in the following way:*

- *the set* $AL_{ESTW}(\mathbf{T}) = V_{a_{id}} \cup V_{a_t} \cup \mathbb{Z}_1 \cup \{ \text{“,”} \}$ *is called an alphabet of language* $ESTW(\mathbf{T})$*,*
- *the set of atomic formulas of the language* $ESTW(\mathbf{T})$ *is defined as a set of four-element tuples in the following form:* (i, b, l, s)*, where* $i \in V_{a_{id}}$*,* $b \in V_{a_t}$ *and* l, s *are integer numbers such that* $l > 1$ *and* $s > 0$*.*

Now, we define a semantics of the language $ESTW(\mathbf{T})$. The formulas of the language $ESTW(\mathbf{T})$ are treated as descriptions of sequences of time windows occurring one after another in system $\mathbf{T}$, and at the same time each two neighboring windows of such a sequence overlap at the connecting points of these windows.

Definition 39. *Let* $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ *be a c-temporal information system and* l *is a length of time windows. The satisfiability of formula* $\phi = (i, b, l, s) \in ESTW(\mathbf{T})$ *(where* $i \in V_{a_{id}}$*,* $b \in V_{a_t}$ *and* l, s *are integer numbers such that* $l > 1$ *and* $s > 0$*) by a time window* $W = (u_1, ..., u_n) \in TW(\mathbf{T})$ *(denoted by* $W \models_{ESTW(\mathbf{T})} \phi$*), is defined in the following way:*

$W \models_{ESTW(\mathbf{T})} (i, b, l, s) \Leftrightarrow$

$$\forall_{j \in \{1,...,n\}}\ a_{id}(u_j) = i \ \wedge \ l = n \ \wedge p \ mod \ (l-1) = 0 \ \wedge \frac{p}{l-1} < s$$

where $p = card(\{x \in U : x \ precedes \ u_1 \ \wedge \ b \leq_{a_t} a_t(x)\})$*.*

Let us notice that a time window $W = (u_1, ..., u_n) \in TW(\mathbf{T})$ satisfies a formula $\phi = (i, b, l, s) \in ESTW(\mathbf{T})$ iff the following four conditions are satisfied:

1. the time window W describes parameters of the object with the identifier i,
2. the size of the time window W is equal l,
3. the integer number of time windows with the length l has been registered since b to $a_t(u_1)$, i.e., the number of time windows with the first point registered not earlier than b and preceding time point u_1 (in the sense of Definition 19) is divisible by $l - 1$,
4. the number of time windows with the length l and with the first time point registered since b to $a_t(u_1)$ is less than s.

The formulas of the language $ESTW$ describe sets of time windows which we call sequences of time windows.

Definition 40 (*A sequence of time windows*). *Let* $\mathbf{T} = (U,\ A,\ a_{id},\ \leq_{a_{id}},\ a_t,\ \leq_{a_t})$ *be a c-temporal information system.*

1. *Each set of time windows* $|\phi|_{ESTW(\mathbf{T})}$ *which is a meaning of a certain formula* $\phi \in ESTW(\mathbf{T})$ *we call a sequence of time windows in c-temporal information system* $\mathbf{T}$.
2. *A family of all sequences of time windows of a given c-temporal information system* $\mathbf{T}$ *we denote* $SEQ(\mathbf{T})$.
3. *If* $S \in SEQ(\mathbf{T})$ *then a number* $card(S)$ *we call a length of the sequence of time windows* S *and it is denoted by* $Length(S)$.
4. *A family of all sequences of time windows of the length* s *of a given c-temporal information system* $\mathbf{T}$ *is denoted by* $SEQ(\mathbf{T}, s)$.
5. *A family of all sequences of time windows of the length* s *of a given c-temporal information system* $\mathbf{T}$ *that they are windows of the length* l *we denote* $SEQ(\mathbf{T}, l, s)$.
6. *Due to the definition of the language* $ESTW(\mathbf{T})$ *the elements of each sequence of time windows* $S \in SEQ(\mathbf{T}, l, s)$ *are linearly ordered by relation* $\leq_{a_t}$, *then each sequence of time windows sequence may be treated as an ordered time sequence* $S = (W_1, ..., W_s)$ *of time windows from the set* $TW(\mathbf{T}, l)$. *Additionally, each i-th time window of sequence* S *we denote by* $S[i]$, *where* $i \in \{1, ..., s\}$.
7. *Any sequence of time windows* $S' = (W_i, ..., W_j) \in SEQ(\mathbf{T}, j-i+1)$ *created by removing from the sequence* $p = (W_1, ..., W_k) \in SEQ(\mathbf{T}, k)$ *time windows* $W_1, ..., W_{i-1}$ *and* $W_{j+1}, ..., W_k$, *where* $i, j \in \{1, ..., k\}$ *and* $i < j$, *is called a sub-sequence of the sequence* S *and is denoted by* $Subsequence(S, i, j)$.

Here is an example of extracting a sequence of time windows from the c-temporal information system.

Example 26. Let us consider system $\mathbf{T} = (U,\ A,\ a_{id},\ \leq_{a_{id}},\ a_t,\ \leq_{a_t})$ whose objects represent states of vehicles at different time points. Attributes from set A describe sensor parameters of the vehicle at individual points (*e.g.*, velocity, acceleration, location, lane). The distinguished attribute a_{id} is a unique identifier of each vehicle and attribute a_t represents the observation time registered in a given object in system $\mathbf{T}$. Let us assume, for the sake of simplification, that attribute values a_t and a_{id} are natural numbers. Let us take as an example a vehicle marked with the identifier 5 for which a hundred time points are registered in system $\mathbf{T}$ from the time point marked with the identifier 1 to the time point marked with the identifier 100. For such a vehicle a sequence of time windows may be extracted and defined by formula $(5, 1, 4, 3)$ which represents three time windows: W_1, W_2, W_3 defined by formulas $(5, 1, 4)$, $(5, 4, 4)$, $(5, 7, 4)$ (see Fig. 31). $\square$

In practical applications the partition of a time window is an important notion concerning the sequence of time windows. It concerns the situation in which we extract the sequence of time windows from a given time window.

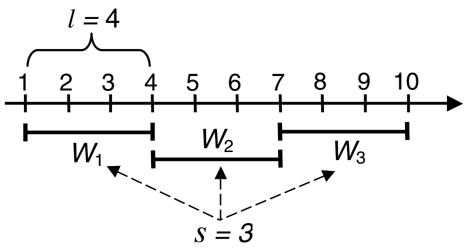


Fig. 31. A sequence of time windows

Definition 41 (*A partition of a time window*). *Let* $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ *be a c-temporal information system and* $W \in TW(\mathbf{T}, m)$. *A family of time windows* $Partition(W, k) = \{W_1, ..., W_k\}$ *is called k-partition of time windows* W*, if the following conditions are satisfied:*

1. $\{W_1, ..., W_k\} \in SEQ(\mathbf{T}, l, k)$, *where* $k \cdot (l - 1) + 1 = m$,
2. $W = \bigcup_{i \in \{1,...,k\}} W_i$.

According to the above definition for a given time window of duration m we can determine k-partition only when $k \cdot (l - 1) + 1 = m$ that is $\frac{m-1}{k} = l - 1$, where l is the length of each of time windows of the obtained partition. Therefore, in order for us to be able to determine k-partition for a given time window of the length m, number m reduced by l has to be divisible by k, and the result of this division is the length of the window in the partition reduced by l. Going back to Example 26 let us notice that it is possible to determine k-partition for $k = 3$ for window $(5, 1, 10)$ (where $m = 10$), and this is the partition into 3 windows, each of length $l = 4$. They are windows: $(5, 1, 4)$, $(5, 4, 4)$ and $(5, 7, 4)$. On the other hand, for the same window $(5, 1, 10)$ we cannot determine k-partition for $k = 4$ because number $m - l = 9$ is not divisible by 4.

Let us notice that it is easy to construct an algorithm which for a given time window $W \in TW(\mathbf{T}, m)$ determines the partition $Partition(W, k)$ through linear overview of window W in order to create k-time windows of this partition.

6.16 Algorithm of Replacing a Sequence of Time Windows with the Sequence of Nodes of a Behavioral Graph

Each time window could be classified into a given temporal concept with the use of stratifying classifier. Each such concept corresponds to one node of the behavioral graph of complex object **G**. Hence, each sequence of time windows may be replaced with the sequence of nodes of behavioral graph **G**. The problem of classification conflict of a given time window consisting in the fact that a time window may be classified into many concepts, could be solved by choosing a concept for which classification certainty is the highest. In other words during classification the concept whose result layer of classification is possibly the

Algorithm 6.8. Replacing a sequence of time windows with the sequence of nodes of a behavioral graph ($MakeTimePath$)

```
Input: A sequence of time windows $S = \{W_1, ...., W_k\} \in SEQ(\mathbf{T}, l, k)$
Output: The path $P \in PATH(\mathbf{G})$ corresponding to the sequence $S$

1 Procedure $MakeTimePath(S)$
2 begin
3     Create empty list $P$
4     for $i := 1$ to $k$ do
5         Select $v_{max} \in V$ such that $\forall v \in V : \mu_v^H(W_i) \leq_H \mu_{v_{max}}^H(W_i)$
6         Add $v_{max}$ to the end of the list $P$.
7     end
8     return $P$
9 end
```

highest is chosen. We present an algorithm performing such an operation (see Algorithm 6.8). However, we assume that during execution of the Algorithm 6.8 the following data structures and algorithms are available:

- $\mathbf{T}$ is a temporal information system,
- $\mathbf{G} = (V, E)$ is a behavioral graph such that $V = \{v_1, ..., v_m\}$,
- $(H, \leq_H)$ is a linearly ordered set such that H is a set of labels of concepts layers and $\leq_H$ is a relation of linear order on the set H,
- $\mu_{v_1}^H, ..., \mu_{v_m}^H$ is a family of stratifying classifiers, which are constructed for temporal concepts corresponding to nodes $v_1, ..., v_m \in V$ (see Algorithm 6.5).

Assuming that all operations of classifiers μ_v^H (for $v \in V$) applications occurring in the above algorithm are executed at the constant time, then the temporal computational complexity of the above algorithm is of order $O(k \cdot m)$, where k is the length of the sequence of time windows which is replaced and $m = card(V)$.

6.17 Temporal Concept for Structured Objects

Complex behaviors of structured objects could be defined on sequences of time windows with the use of complex concepts which are called temporal concepts for structured objects.

We assume that temporal concepts for structured objects are specified by a human expert and are usually used in queries about the status of some structured objects in a particular sequence of time windows. Answers to such queries can be of the form Yes, No or *Does not concern*.

Intuitively each such temporal concept (defined on a sequence of time windows) depends on whether there occurred behaviors defined by temporal concepts for unstructured objects in the observed time windows, with those objects being parts of structured objects. It is usually possible to provide an ontology which shows such a dependence.

For example, temporal concept for structured objects which is a group of two objects (vehicles) *unhurried driving vehicle A after vehicle B in the right lane*, depends on such temporal concepts as: *driving at stable speed*, *changing lanes at constant speed*, *maintaining constant distance between vehicles A and B*, and others.

Temporal concepts for structured objects cannot be straightforwardly approximated by temporal concepts for single time windows because they concern a sequence of time windows, that is, a longer period of time than temporal concepts for time windows.

Therefore, in order to define attributes approximating temporal concepts for structured objects we need to introduce another language which enables us to transfer properties of time windows onto the level of sequences of time windows.

Due to the fact that each sequence of time windows can be replaced with a temporal path in the behavioral graph (see Section 6.16), while describing the properties of time windows we can use temporal paths corresponding to them. In this way, the language FTP below is constructed.

Definition 42 (*A language for defining features of temporal paths*)**.** *Let* $\mathbf{G} = (V, E)$ *be a behavioral graph of complex objects of the fixed type* T*. A language for defining features of temporal paths of behavioral graph* $\mathbf{G}$ *(denoted by* $FTP(\mathbf{G})$ *or* FTP*-language, when* $\mathbf{G}$ *is fixed) is defined in the following way:*

- *the set* $AL_{FTP}(\mathbf{G}) = V \cup \{$ *ExistsNode, EachNode, MajorityNodes, MinorityNodes, FirstNode, LastNode, OrderNodes* $\} \cup \{\neg, \vee, \wedge\}$ *is an alphabet of the language* $FTP(\mathbf{G})$*,*
- *for any* $v, v' \in V$ *expressions of the form:* $ExistsNode(v)$*,* $EachNode(v)$*,* $MajorityNodes(v)$*,* $MinorityNodes(v)$*,* $FirstNode(v)$*,* $LastNode(v)$*,* $OrderNodes(v, v')$ *are atomic formulas of the language* $FTP(\mathbf{G})$*.*

Presently, we define the semantics of the language $FTP(\mathbf{G})$. The formulas of the language $FTP(\mathbf{G})$ express properties of paths in the behavioral graph $\mathbf{G}$. For example, formula $ExistsNode(v)$ is interpreted as description of all those paths in the behavioral graph in which there exists node v.

Definition 43. *Let* $\mathbf{G} = (V, E)$ *be a behavioral graph of complex objects of the fixed type* T*. The satisfiability of an atomic formula* $\phi \in FTP(\mathbf{G})$ *by a temporal path* $P = (v_1, \ldots, v_k) \in PATH(\mathbf{G})$ *(denoted by* $P \models_{FTP(\mathbf{G})} \phi$*), is defined in the following way:*

1. $P \models_{FTP(\mathbf{G})} ExistsNode(v) \Leftrightarrow \exists_{i \in \{1,\ldots,k\}}\ v_i = v,$
2. $P \models_{FTP(\mathbf{G})} EachNode(v) \Leftrightarrow \forall_{i \in \{1,\ldots,k\}}\ v_i = v,$
3. $P \models_{FTP(\mathbf{G})} MajorityNodes(v) \Leftrightarrow$
$$card(\{i \in \{1, ..., k\} : v_i = v\}) > \lfloor (k-1)/2 \rfloor,$$
4. $P \models_{FTP(\mathbf{G})} MinorityNodes(v) \Leftrightarrow$
$$card(\{i \in \{1, ..., k\} : v_i = v\}) < \lceil (k-1)/2 \rceil,$$

5. $P \models_{FTP(\mathbf{G})} FirstNode(v) \Leftrightarrow v_1 = v$*,*
6. $P \models_{FTP(\mathbf{G})} LastNode(v) \Leftrightarrow v_k = v$*,*
7. $P \models_{FTP(\mathbf{G})} OrderNodes(v, v') \Leftrightarrow \exists_{i,j \in \{1,\dots,k\}}\ i < j\ \wedge\ v_i = v\ \wedge\ v_j = v'$*.*

Below, we provide a few examples of the FTP-language formulas.

– If node v of a certain behavioral graph of a single vehicle corresponds to the concept of *stable speed in the right lane*, then formula $EachNode(v)$ describes a temporal path in which the vehicle drives the whole time at the stable speed in the right lane.
– If node v of a certain behavioral graph of a single vehicle corresponds to the concept of *accelerating in the right lane*, then formula $ExistsNode(v)$ describes a temporal path in which the vehicle for a certain amount of time accelerates.
– If nodes v_1 and v_2 of a certain behavioral graph of a single vehicle correspond respectively to the concepts of *stable velocity in the right lane* and *accelerating in the right lane*, then formula $MajorityNodes(v_1) \wedge ExistsNode(v_2)$ describes a temporal path in which for most of the time the vehicle drive at the stable speed but for some time the vehicle increases speed.

Formulas of the FTP language can be used for defining temporal path features in such a way that to each formula a temporal pattern for temporal paths can be attached.

Definition 44 (*A temporal pattern for temporal paths*)**.** *Let* $\mathbf{G} = (V, E)$ *be a behavioral graph of complex objects of the fixed type* T*. Each formula of the language* $FTP(\mathbf{G})$ *is called a temporal pattern for temporal paths in the behavioral graph* $\mathbf{G}$*.*

We assume that any temporal pattern ought to be defined by a human expert using domain knowledge accumulated for the given complex dynamical system.

6.18 Information System of Temporal Paths

The properties of accessible temporal paths may be represented in special information system which is called *an information system of temporal paths.*

Definition 45 (*An information system of temporal paths*)**.** *Let us assume that:*

– $\mathbf{G} = (V, E)$ *is a behavioral graph of complex objects of the fixed type,*
– s *is a fixed length of temporal paths* ($s > 1$)*,*
– $\Phi = \{\phi_1, ..., \phi_k\} \subseteq FTP(\mathbf{G})$ *is a defined by experts family of temporal patterns for temporal paths (sub-language of the language* $FTP(\mathbf{G})$*),*
– $\mathcal{P}_{FTP} = (\overline{U}, \Phi, \models_{FTP(\mathbf{G})})$ *is a property system, where* $\overline{U} \subseteq PATH(\mathbf{G}, s)$*.*

The information system $\overline{\mathbf{G}} = (\overline{U}, \overline{A})$ *defined be the property system* $\mathcal{P}_{FTP}$ *is called an information system of temporal paths (*TP*-information system).*

Evidently, construction of system $\overline{\mathbf{G}}$ requires generating a family of temporal paths of a fixed length. It is possible to construct an algorithm generating all temporal paths of the established duration s from a given behavioral graph $\mathbf{G} = (V, E)$ which would work in such a way that for each node $v \in V$ there would be generated all paths of the established duration s which start in the node. If the number of edges coming out of a given node of graph $\mathbf{G}$ equaled no more than r, then the number of paths generated using such an algorithm would be pessimistically equal $card(V) \cdot r^{s-1}$, which in practice may be a great number of paths. For example, if $card(V) = 10$, $r = 4$ and $s = 10$ then the number of paths to generate equals over 2.6 million! Therefore, in practice, during system $\overline{\mathbf{G}}$ construction only chosen paths from the set $PATH(\mathbf{G}, s)$ are generated.

Although the set $PATH(\mathbf{G}, s)$ may be sampled, it is reasonable to assume that in system $\overline{\mathbf{G}}$ only those paths are taken into account which occur in the training data. This means that in system $\overline{\mathbf{G}}$ only such a path exists that is registered in the data for the specific complex object. Obviously, the node sequence is not directly registered in the data, but time point sequence is registered which may be grouped into time windows and then into sequences of time windows. These sequences, thanks to classifiers constructed for individual temporal concepts, may be changed into the node sequence from set V informing about the membership of time windows to individual temporal concepts. The set of all temporal paths observed in data of duration s we denote as $DPATH(\mathbf{G}, s)$. For small behavioral graphs the size of the set $DPATH(\mathbf{G}, s)$ is comparable with the size of the set $PATH(\mathbf{G}, s)$. However, for nontrivially small behavioral graphs the size of the set $DPATH(\mathbf{G}, s)$ is much smaller than the size of the set $PATH(\mathbf{G}, s)$. We present the algorithm of generating the set $DPATH(\mathbf{G}, s)$ (see Algorithm 6.9).

Algorithm 6.9. Generating temporal paths observed in data

```
Input:
  – temporal information system $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ such that
    $U = \{u_1, ..., u_n\}$,
  – behavioral graph $\mathbf{G} = (V, E)$ such that $V = \{v_1, ..., v_m\}$,
  – fixed length of temporal paths $s$ (where $s > 1$),
  – fixed length of time windows $l$ (where $l > 1$).

Output: The set $DPATH(\mathbf{G}, s)$
1 begin
2     Generate set $TW := TW(\mathbf{T}, s \cdot (l-1) + 1)$.
3     Create empty list of paths $DPATH$.
4     for all $W \in TW$ do
5         $path := MakeTimePath(Partition(W, s))$
6         Add $path$ to the list $DPATH$.
7     end
8     return $DPATH$
9 end
```

In regard to determining the set $TW(\mathbf{T}, s \cdot (l-1)+1)$ and executions of procedures *MakeTimePath* and *Partition*, the pessimistic time complexity of the Algorithm 6.9 is of order $O(n \cdot log\ n + \ n \cdot s \cdot l + n \cdot s \cdot m)$, where n is the number of objects in the set U, s is the length of temporal paths, l is the length of time windows and $m = card(V)$.

Example 27. For the behavior graph from Fig. 28 an information system of temporal paths may be constructed. In order to do this one may use temporal patterns for temporal paths chosen from the following collection of patterns:

1. the vehicle accelerates (decelerates, maintains stable speed) in the right lane at the first node (at the last node, at some node, at all nodes, at the minority of nodes, at the majority of nodes) of the temporal path,
2. the vehicle accelerates and at the same time changes the lane from the right to the left one at the first node (at the last node, at some node, at all nodes, at the minority of nodes, at the majority of nodes) of the temporal path,
3. the vehicle decelerates (maintains stable speed) and at the same time changes lanes from the left to the right one at the first node (at the last node, at some node, at all nodes, at the minority of nodes, at the majority of nodes) of the temporal path,
4. the vehicle accelerates (decelerates , maintains stable speed) in the right lane at some node of the temporal path, and after that the vehicle accelerates (decelerates , maintains stable speed) in the right lane at one of the following nodes,
5. the vehicle accelerates (maintains stable speed) and at the same time changes the lane from the right to the left one at some node of the temporal path, and after that at one of the following nodes the vehicle decelerates (maintains stable speed) in the right lane,
6. the vehicle decelerates (maintains stable speed) and at the same time changes the lane from the left to the right one at some node of the temporal path, and after that at one of the following nodes the vehicle decelerates (maintains stable speed) in the right lane.

It is easy to notice that each of the above patterns may be expressed in language *FTP*. □

In approximating temporal concepts for structured objects (see Section 6.20), it is necessary to use information systems of temporal paths which are constructed on the basis of behavioral graphs illustrating relation changes between parts of structured objects. The example below shows how such an information system for behavior graph from Fig. 30 may be constructed.

Example 28. The behavioral graph from Fig. 30 describes distance changes between parts of the structured object which is a pair of vehicles driving on the road. For this graph the information system of temporal paths may be constructed using temporal patterns for temporal paths chosen from the following collection of patterns:

1. the distance between both vehicles is short and is decreasing (is increasing, is stable) at the first node (at the last node, at some node, at all nodes, at the minority of nodes, at the majority of nodes) of the temporal path,
2. the distance between both vehicles is long and is decreasing (is increasing, is stable) at the first node (at the last node, at some node, at all nodes, at the minority of nodes, at the majority of nodes) of the temporal path. □

The choice of specific temporal patterns to construct the information system of temporal paths depends on the temporal concept which is to be approximated using this system, obviously after performing the grouping (clustering) of temporal paths (see Section 6.20).

6.19 Clustering Temporal Paths

The properties of temporal paths expressed using temporal patterns may be used to approximate temporal concepts for structured objects. Frequently, however, objects of system $\overline{\mathbf{G}}$ are not yet suitable for their properties to describe temporal concepts for structured objects. It happens that way because there are too many of these objects and the descriptions of temporal paths, which they represent, are too detailed. Hence, if applied for temporal concept approximation for structured objects, the extension of the created classifier would be too small, that is, the classifier could classify too small number of tested paths.

Therefore, similarly to the case of temporal concepts for time windows the object clustering is applied which leads to obtaining the family of clusters of temporal paths. The grouping tools are the same as in the case of time window grouping. Therefore, to define path clusters we use the language $ECTP$ which similarly to the language $ECTW$ is based on the language NL.

Definition 46 (*A language for extracting clusters of temporal paths*)**.** *Let us assume that:*

– $\mathbf{G} = (V, E)$ *is a behavioral graph of complex objects of the fixed type,*
– $\overline{\mathbf{G}} = (\overline{U}, \overline{A})$ *is an information system of temporal paths of behavioral graph* $\mathbf{G}$*.*

Any neighborhood language $NL(\overline{\mathbf{G}})$ *we call a language for extracting clusters of temporal paths from behavioral graph* $\mathbf{G}$ *(*$ECTP$*-language) and we denote it by* $ECTP(\mathbf{G})$*.*

Formulas of the language $ECTP$ enable to define clusters of temporal paths from the behavioral graph.

Definition 47 (*A cluster of temporal paths*)**.** *Let* $\mathbf{G} = (V, E)$ *be a behavioral graph of complex objects of the fixed type. We call the meaning of each formula* $\phi \in ECTP(\mathbf{G})$ *a cluster of temporal paths of behavioral graph* $\mathbf{G}$*.*

In the example below we illustrate how the dissimilarity function may be defined for grouping temporal paths.

Example 29. For the information system of temporal paths obtained with the use of temporal patterns from Example 27, the dissimilarity function may be constructed on the basis of expert knowledge. In order to do this, the dissimilarity function value should be obtained from experts for a certain, i.e., the most representative set of object pairs of this system. It should be also determined what temporal concept is approximated using clusters obtained with the help of the dissimilarity function being defined (see Section 6.20). It is possible to approximate concepts defined on the set of unstructured objects as well as on the set of structured objects. Let us assume that this is the concept concerning the overtaking maneuver of vehicle B by vehicle A and we mean this fragment of the overtaking maneuver when vehicle B drives in the right lane and vehicle A while driving behind vehicle B changes the lane from the right to the left one. In terms of this concept and knowledge about typical behaviors of vehicle A, the expert may instantly define several groups of vehicles which, if put in place of vehicle A, behave very similarly, that is, vehicles which are in one of such groups in terms of the dissimilarity function which is being constructed, should not differ significantly. For example, they may be the following groups of vehicles:

I. vehicles which at the beginning of the temporal path drive in the right lane and then drive off the right lane into the left one,
II. vehicles which along the whole temporal path drive in the right lane,
III. vehicles which along the temporal path drive in the left lane,
IV. vehicles which at the beginning of the temporal path drive in the left lane and then drive off the left lane into the right one.

Let us now consider five specific vehicles which behave in the following way:

1. vehicle u_1 accelerates in the right lane at the beginning of the temporal path, and then still accelerates and commences changing lanes from the right to the left one, but at the end of the temporal path is still not in the left lane,
2. vehicle u_2 drives at a stable speed in the right lane at the beginning of the temporal path, and then commences changing lanes from the right to the left one at a stable speed, but at the end of the temporal path it is still not in the left lane,
3. vehicle u_3 drives at a stable speed in the right lane along the whole temporal path,
4. vehicle u_4 drives at a stable speed in the left lane at the beginning of the temporal path, and then decelerates in the left lane,
5. vehicle u_5 drives at a stable speed in the left lane at the beginning of the temporal path, and then at a stable speed commences changing lanes from the left to the right one, but at the end of the temporal path it is still not in the right lane.

In Table 6 values of chosen temporal patterns for objects u_1, u_2, u_3, u_4 and u_5 are presented.

It is evident that object u_1 belongs to the group of vehicles I and it does not belong to groups II, III and IV. Similarly, object u_2 also belongs to group I. The

Table 6. An exemplary objects of some *TP*-information system

No	**Temporal patterns for temporal paths**	u_1	u_2	u_3	u_4	u_5
1.	The vehicle accelerates driving in the right lane at the first node of the temporal path	1	0	0	0	0
2.	The vehicle drives at a stable speed in the right lane at the first node of the temporal pattern	0	1	1	0	0
3.	The vehicle accelerates and changes lanes from the right to the left one at some node of the temporal path	1	0	0	0	0
4.	The vehicle drives at a stable speed and changes lanes from right to the left one at some node of the temporal path	0	1	0	0	0
5.	The vehicle drives at a stable speed in the left lane at the first node of the temporal path	0	0	0	1	1
6.	The vehicle decelerates in the left lane at some node of the temporal path	0	0	0	1	0
7.	The vehicle drives at a stable speed and changes lanes from the left to the right one at some node of the temporal path	0	0	0	0	0
8.	The vehicle decelerates and changes lanes from the left to the right one at some node of the temporal path	0	0	0	0	1
9.	The vehicle drives at a stable speed in the right lane at the last node of the temporal path	0	0	1	0	0
10.	The vehicle drives at a stable speed in the left lane at the last node of the temporal path	0	0	0	0	0

reason for this is the fact that both objects drive in the right lane first and then drive off to the left lane, but object u_2 drives at a stable speed the whole time and object u_1 accelerates. Therefore, the dissimilarity function value for these two objects is low, that is, 0.1, whereas object u_3 belongs to group II because it drives in the right lane the whole time. That is why its behavior differs significantly from the behavior of object u_2, for object u_3 only drives in the right lane while object u_2 changes lanes from the right to the left one. Hence, the dissimilarity function value for objects u_2 and u_3 is larger and equals 0.3. Object u_4 belongs to group III (drives only in the left lane) and therefore its behavior has even less in common with the behavior of object u_2. Therefore, the dissimilarity function value for the pair of objects u_2 and u_4 is 0.4. Finally, object u_5 belongs to group IV and its behavior differs the most from the behavior of object u_1. The reason for this is the fact that object u_5 not only commences driving in the left lane (unlike u_1, which commences driving in the right lane), but at the end of the considered temporal path u_5 changes lanes from the left to the right one, unlike u_1, which changes lanes from the right to the left one. Moreover, the object u_5

Table 7. Values of dissimilarity function for pairs of temporal paths

	u_1	u_2	u_3	u_4	u_5
u_1	0	0.1	0.4	0.4	1.0
u_2	0.1	0	0.3	0.3	0.9
u_3	0.4	0.3	0	0.6	0.4
u_4	0.4	0.4	0.6	0	0.4
u_5	1.0	0.9	0.4	0.4	0

decelerates, while the object u_1 accelerates. Therefore, the dissimilarity function value for the pair of objects u_1 and u_5 is the highest possible and equals 1.0. Let us notice that the dissimilarity function value for the pair of objects u_2 and u_5 is bit smaller (0.9) because the object u_5 decelerates, while the object u_2 maintains stable speed. The dissimilarity function values for all pairs of objects proposed by the expert are put together in Table 7. Finally, let us notice that objects u_1, u_3, u_4 and u_5 may be treated as standards for which the deviation is 0.1. $\square$

Now, to construct a new information system which represents the features of clusters of temporal paths we need to define a new language.

Definition 48 (*A language for definnig features of clusters of temporal paths*). *Let us assume that:*

- $\mathbf{G} = (V, E)$ *is a behavioral graph of complex objects of the fixed type,*
- $\overline{\mathbf{G}} = (\overline{U}, \overline{A})$ *is a information system of temporal paths for behavioral graph* $\mathbf{G}$.

A language for definnig features of clusters of temporal paths for behavioral graph $\mathbf{G}$ *(denoted by* $FCTP(\mathbf{G})$ *or* $FCTP$*-language, when* $\mathbf{G}$ *is fixed) is defined in the following way:*

- *the set* $AL_{FCTP}(\mathbf{G}) = AL_{FTP}(\mathbf{G}) \cup \{ExistsPath, EachPath, MajorityPaths, MinorityPaths\}$ *is an alphabet of the language* $FCTP(\mathbf{G})$,
- *for any* $\alpha, \beta \in FTP(\mathbf{G})$ *expressions of the form:* $ExistsPath(\alpha)$, $EachPath(\alpha)$, $MajorityPaths(\alpha)$, $MinorityPaths(\alpha)$ *are atomic formulas of the language* $FCTP(\mathbf{G})$.

Now, we determine semantics of the language $FCTP(\mathbf{G})$. The formulas of the language $FCTP(\mathbf{G})$ we interpret as a description of clusters of temporal paths of graph G. For example, formula $ExistsPath(\alpha)$ is interpreted as the description of all those clusters of temporal paths in which there exists at least one path satisfying formula α.

Definition 49. *Let* $\mathbf{G} = (V, E)$ *be a behavioral graph of complex objects of the fixed type* T. *The satisfiability of an atomic formula* $\phi \in FCTP(\mathbf{G})$ *by a family of temporal paths* $\mathcal{P} = \{P_1, ..., P_k\} \subseteq PATH(\mathbf{G})$ *(denoted by* $\mathcal{P} \models_{FCTP(\mathbf{G})} \phi$*) is defined in the following way:*

1. $\mathcal{P} \models_{FCTP(\mathbf{G})} ExistsPath(\alpha) \Leftrightarrow \exists_{P \in \mathcal{P}} P \models_{FTP(\mathbf{G})} \alpha$,
2. $\mathcal{P} \models_{FCTP(\mathbf{G})} EachPath(\alpha) \Leftrightarrow \forall_{P \in \mathcal{P}} P \models_{FTP(\mathbf{G})} \alpha$,
3. $\mathcal{P} \models_{FCTP(\mathbf{G})} MajorityPaths(\alpha) \Leftrightarrow$

$$card(\{P \in \mathcal{P} : P \models_{FTP(\mathbf{G})} \alpha\}) > \lfloor (k-1)/2 \rfloor,$$

4. $\mathcal{P} \models_{FCTP(\mathbf{G})} MinorityPaths(\alpha) \Leftrightarrow$

$$card(\{P \in \mathcal{P} : P \models_{FTP(\mathbf{G})} \alpha\}) < \lceil (k-1)/2 \rceil.$$

The patterns of the language $FCTP$ may be applied to define the features of clusters of temporal paths.

The example below illustrates how one may define properties of clusters of temporal paths which are constructed using the dissimilarity function from Example 29.

Example 30. Using the dissimilarity function from Example 29, the objects from the information system of temporal paths from Example 27 may be grouped. However, in order to construct an information system for the determined family of clusters, patterns should be established which serve computing features of clusters of temporal paths. For example, they may be the following patterns:

1. in each temporal path of the cluster vehicles first drive in the right lane and then change lanes from the right to the left one (the pattern describes the property of vehicles from group I from Example 29),
2. in each temporal path of the cluster vehicles drive only in the right lane (the pattern describes the property of vehicles from group II from Example 29),
3. in each temporal path of the cluster vehicles first drive in the left lane and then change lanes from the left to the right one (the end of the overtaking maneuver),
4. in each temporal path of the cluster there exists a node at which vehicles drive in the left lane,
5. in the majority of temporal paths of the cluster there occurs no deceleration,
6. in the majority of temporal paths of the cluster vehicles drive at a stable speed the whole time.

It is easy to notice that each of the above patterns may be expressed in language $FCTP$. □

The choice of specific patterns to construct the information system of the temporal path clusters depends on the concept which is to be approximated with the help of this system (see Section 6.20).

Clusters of temporal paths are represented by the information systems which we call *an information system of clusters of temporal paths.*

Definition 50 (*An information system of clusters of temporal paths*)**.** *Let us assume that:*

– $\mathbf{G} = (V, E)$ *is a behavioral graph of complex objects of the fixed type,*
– $\psi_1, ..., \psi_n \in ECTP(\mathbf{G})$ *is a defined by experts family of temporal patterns corresponding to clusters of temporal paths* $CL_1, ..., CL_n$*,*

– *$\Phi = \{\phi_1, ..., \phi_m\} \subseteq FCTP(\mathbf{G})$ is a defined by experts family of features of clusters of temporal paths,*
– *$\mathcal{P}_{FCTP} = (\overline{\overline{U}}, \Phi, \models_{FCTP(\mathbf{G})})$ is a property system, where $\overline{\overline{U}} = \{CL_1, ..., CL_n\}$.*

The information system $\overline{\overline{\mathbf{G}}} = (\overline{\overline{U}}, \overline{\overline{A}})$ defined be the property system $\mathcal{P}_{FCTP}$ is called an information system of clusters of temporal paths (CTP-information system).

Evidently, constructing system $\overline{\overline{\mathbf{G}}}$ requires generating clusters of temporal paths in such a way that it would be possible to check the ability to satisfy the formulas of the language $FCTP(\mathbf{G})$. Such clusters may be generated by the linear search in the table $\overline{\mathbf{G}}$ and assigning objects of this table to individual clusters.

Algorithm 6.10 presented bellow determines for a given temporal path the list of feature values of the cluster to which the tested temporal path belongs. Therefore, this algorithm is important for testing temporal paths. We assume

Algorithm 6.10. Computation features of cluster to which the tested temporal path belongs

Input: temporal path $P = (v_1, \ldots, v_s) \in PATH(\mathbf{G})$.
Output: The list of feature values of the cluster to which the tested temporal path P belongs

```
1  Procedure GetClusterRow(P)
2  begin
3      Create empty list row
4      for $\phi_1$, ..., $\phi_m$ do
5          if $P \models_{FTP(\mathbf{G})} \phi_i$ then
6              Add '1' to the end of the list row
7          else
8              Add '0' to the end of the list row
9          end
10     end
11     Create new object $u_{row}$ of the system $\overline{\mathbf{G}}$ on the basis values of
       attributes from the list row.
12     Select a formula $\psi = (std, \varepsilon) \in \{\psi_1, ..., \psi_n\}$ such that:
13       1. $u_{row} \models_{FCTP(\mathbf{G})} \psi$ and
14       2. $DISM_{\overline{\mathbf{G}}}(u_{row}, std) = min_{i \in \{1,...,n\}}\{DISM_{\overline{\mathbf{G}}}(u_{row}, std_i)\}$
15     Select object $u_\psi \in \overline{\overline{U}}$ corresponding to the formula $\psi$
16     Create empty list clusterRow
17     for $i = 1$ to $k$ do
18         $clusterRow := clusterRow + \overline{\overline{a}}_i(u_\psi)$
19     end
20     return clusterRow
21 end
```

that during execution of the Algorithm 6.10 the following data structures are available:

- a behavioral graph $\mathbf{G} = (V, E)$ of complex objects of a fixed type,
- a fixed length s of temporal paths in the behavioral graph $\mathbf{G}$,
- a family of temporal patterns $\phi_1, ..., \phi_m \in FTP(\mathbf{G})$) and a system $\overline{\mathbf{G}} = (\overline{U}, \overline{A})$ such that attributes from the set $\overline{A}$ correspond to formulas $\phi_1, ..., \phi_m$,
- a dissimilarity function $DISM_{\overline{\mathbf{G}}}$ of object pairs from the system $\overline{\mathbf{G}}$, approximated using the stratifying classifier $\mu_{DISM_{\overline{\mathbf{G}}}}$,
- a family of temporal patterns $\psi_1, ..., \psi_n \in ECTP(\mathbf{G})$ such that $\psi_i = (std_i, \varepsilon_i)$ (for $i = 1, ..., n$) and a system $\overline{\overline{\mathbf{G}}} = (\overline{\overline{U}}, \overline{\overline{A}})$ such that objects from the set $\overline{\overline{U}}$ correspond to clusters described by formulas $\psi_1, ..., \psi_n$ and $\overline{\overline{A}} = \{\overline{\overline{a}}_1, ..., \overline{\overline{a}}_k\}$.

Assuming that all operations of checking the satisfiability of formulas for a given path and computing values of dissimilarity function (occurring in the above algorithm) are carried out at the constant time, then the temporal computational complexity of the above algorithm is of order $O(m+n+k)$ where m is the number of used temporal path features, n is the number of clusters of temporal paths, and k is the number of features of clusters of temporal paths.

6.20 Temporal Concept Table for Structured Objects

CTP-information systems constructed for different unstructured objects may be joined in order to obtain information systems describing features of structured objects. We obtain objects of such a system by arranging (joining) all possible objects of information systems being joined. From the mathematical point of view such an arrangement is the Cartesian product of sets of objects of joined information systems (see [78, 84, 186, 187]). In this way we obtain an information system that may be used to approximate concepts for structured objects (see Definition 52). However, from the point of view of domain knowledge not all mentioned above object arrangements are possible and sensible. For example, if we approximate the concept of overtaking performed by two vehicles (overtaking and overtaken ones), then it is sensible to link the three following path clusters:

1. the first path cluster (coming from the behavior graph of the overtaking vehicle) describes a sequence of vehicle behaviors which behave in the same way as the overtaking vehicle (*e.g.*, accelerate and change lanes from the right to the left one),
2. the second path cluster (coming from the behavior graph of the overtaken vehicle) describes a sequence of vehicle behaviors which behave in the same way as the overtaken vehicle (*e.g.*, drive with a stable speed in the right lane),
3. the third path cluster (coming from the behavior graph describing relation changes between the overtaking and overtaken vehicle) describes a sequence of behaviors of such vehicles pairs that the relations determined between them change in a way connected with overtaking (*e.g.*, the distance between vehicles decreases rather quickly).

For the above reason, that is, to eliminate object arrangements which are unreal or are not sensible, a relation of constraints is formulated which informs which objects may be arranged in order to obtain positive or negative example of approximated concept (for structured objects) and which cannot be arranged. The relation of constraints we define in the form of the formula of the language *GDL*.

Definition 51 (*An information system of clusters of temporal paths for structured objects*)**.** *Let us assume that:*

- $\mathbf{G}_i = (V_i, E_i)$ *is a behavioral graph of complex objects of the fixed type* T_i, *for* $i = 1, ..., k$,
- $\overline{\overline{\mathbf{G}}}_i = (\overline{\overline{U}}_i, \overline{\overline{A}}_i)$ *is a CTP-information system for behavioral graph* $\mathbf{G}_i$, *for* $i = 1, ..., k$,
- $\mathbf{G}_R = (V_R, E_R)$ *a behavioral graph representing changes of relations between parts of structured objects, where such parts are objects of types* $T_1, ..., T_k$,
- $\overline{\overline{\mathbf{G}}}_R = (\overline{\overline{U}}_R, \overline{\overline{A}}_R)$ *is a CTP-information system for behavioral graph* $\mathbf{G}_R$,
- $\mathbf{G}_\otimes = (U_\otimes, A_\otimes)$ *is an information system, where:*
  - $U_\otimes = \overline{\overline{U}}_1 \times \ldots \times \overline{\overline{U}}_k \times \overline{\overline{U}}_R$,
  - $A_\otimes = A_1 \cup \ldots \cup A_k \cup A_R$, *where attributes from sets* $A_1, ..., A_k, A_R$ *are natural extension of attributes from sets* $\overline{\overline{A}}_1, ..., \overline{\overline{A}}_k, \overline{\overline{A}}_R$ *on the set* $U_\otimes$,
- $\Phi \in GDL(\mathbf{G}_\otimes)$ *is a formula of constraints.*

An information system of clusters of temporal paths for structured objects (SCTP-information system) is an information system $\mathbf{G}_\bowtie = (U_\bowtie, A_\bowtie)$, *where:*

- $U_\bowtie = \{u \in U_\otimes : u \models_{GDL(\mathbf{G}_\otimes)} \Phi\}$,
- $A_\bowtie = A_\otimes$, *i.e., the set* $A_\bowtie$ *contains all attributes from the set* $A_\otimes$, *that are restricted to* $U_\bowtie$.

The notion of information system $\mathbf{G}_\bowtie$ may be used to approximate concepts for structured objects. In order to do this it is sufficient to add a decision attribute to this system which describes the approximated concept. We assume that for each arrangement of objects accepted by constraints (satisfying the formula of constraints), the expert adds the decision value informing about whether a given arrangement belongs to the approximated concept of the higher level or not.

Now, the definition of a temporal concept table for structured objects may be presented.

Definition 52 (*A temporal concept table for structured objects*)**.** *Let us assume that:*

- $\mathbf{G}_\bowtie = (U_\bowtie, A_\bowtie)$ *is an information system of clusters of temporal paths for structured objects,*
- $C \subseteq U_\bowtie$ *is a temporal concept for structured objects.*

A temporal concept table for structured objects for the concept C *is a decision table* $\mathbf{G}^C_\bowtie = (U_\bowtie, A_\bowtie, d_C)$, *where the decision attribute* d_C *is representing membership of objects from the set* $U_\bowtie$ *to the concept* C.

As we already mentioned, the relation of constraints is defined as a formula in the language *GDL* (see Definition 5) on the basis of attributes of the table $\mathbf{G}_{\otimes}$. However, the relation of constraints may also be approximated with the help of classifiers. It is required, in such a case, to give examples of objects belonging and not belonging to this relation, that is, satisfying and not satisfying the formula Φ corresponding to this relation (see [78]).

Example 31. In Fig. 32 there is presented a construction scheme of temporal concept table for structured objects consisting of two vehicles, i.e., an overtaking vehicle (vehicle A) and an overtaken vehicle (vehicle B). Any object of this table is represented by a triple of clusters $(P_i^{(A)}, P_i^{(B)}, P_i^{(R)})$ for $i \in \{1, ..., card(U_{\bowtie})\}$, where:

- $P_i^{(A)}$ denotes the cluster of overtaking vehicles,
- $P_i^{(B)}$ denotes the cluster of overtaken vehicles,
- $P_i^{(R)}$ denotes the cluster of pairs of both vehicles.

Attributes describing the clusters of overtaking and overtaken vehicles, *e.g.*, clusters $P_i^{(A)}$ and $P_i^{(B)}$ are constructed as presented in Example 30. They de-

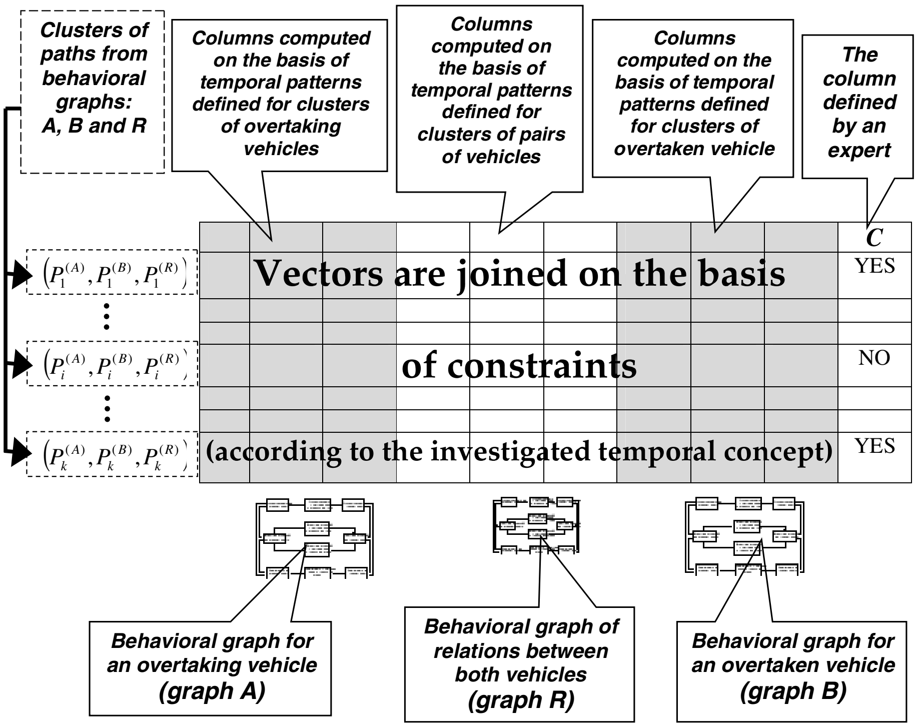

Clusters of paths from behavioral graphs: A, B and R
Columns computed on the basis of temporal patterns defined for clusters of overtaking vehicles
Columns computed on the basis of temporal patterns defined for clusters of pairs of vehicles
Columns computed on the basis of temporal patterns defined for clusters of overtaken vehicle
The column defined by an expert
C
$(P_1^{(A)}, P_1^{(B)}, P_1^{(R)})$
$(P_i^{(A)}, P_i^{(B)}, P_i^{(R)})$
$(P_k^{(A)}, P_k^{(B)}, P_k^{(R)})$
Vectors are joined on the basis
of constraints
(according to the investigated temporal concept)
YES
NO
YES
Behavioral graph for an overtaking vehicle (graph A)
Behavioral graph of relations between both vehicles (graph R)
Behavioral graph for an overtaken vehicle (graph B)

Fig. 32. The scheme of the temporal concept table of structured objects (two vehicles during the overtaking maneuver)

scribe rather general behaviors of individual vehicles observed during the sequence of time windows. Whereas, the attributes describing cluster $P_i^{(R)}$ describe the properties of both vehicles (A and B) in terms of relation changes between these vehicles. During observing vehicles during the overtaking maneuver the most important thing is to observe the distance changes between these vehicles (see Example 28). Therefore, to describe the properties of $P_i^{(R)}$ clusters the following patterns may be applied:

1. in each temporal path of the cluster, at the majority of nodes the distance between A and B is short and is still decreasing,
2. in each temporal path of the cluster there exists a node at which the distance between A and B is long,
3. in each temporal path of the cluster at the majority of nodes the distance between A and B is short and is increasing,
4. in the majority of temporal paths of the cluster there exists a node at which the distance between A and B is long and it is decreasing, and after it there exists a node at which the distance between A and B is short and is decreasing,
5. in each temporal path of the cluster and at all the nodes of these paths the distance between A and B is short.

The information system obtained in such a way may be enriched with the decision attribute which describes the membership of individual clusters of time windows to the established temporal concept for the group of both vehicles (*e.g.*, it may be the concept: *vehicle B drives in the right lane and vehicle A while driving behind vehicle B changes the lane from the right to the left one*). In this way we obtain a decision table which may serve the approximation of this concept.

It is worth noticing that the attribute created on the basis of the last of the above mentioned patterns may be used to construct the constraint formula. If we build the constraint formula based on this attribute, then the clusters joints representing vehicles which are at a long distance are eliminated from the system $\mathbf{G}_{\otimes}$. Such cluster joints are not useful for differentiating pairs of vehicles which drive close to each other and are involved in the overtaking maneuver from those pairs of vehicles which also drive close to each other, but they are not involved in the overtaking maneuver. □

A question arises, how it is possible that the expert may propose the decision value related to the membership to system $\mathbf{G}_{\bowtie}$. To make this decision the expert has at his or her disposal attribute values from the set $A_{\bowtie}$. They are attributes proposed earlier by the expert as features of clusters of temporal paths. The expert is able to propose them in such a way that he/she may now use them successively to determine the membership of the whole clusters to concept C. In other words cluster features must be defined in such a way that they could serve to determine the membership of clusters of time paths to temporal concept C.

If $E = \{e_1, ..., e_l\}$ is a family of layer labels of concept C, then for table $\mathbf{G}_{\bowtie}^{C}$ stratifying classifier μ_C^E may be constructed. This classifier enables classifying structured objects to concept C.

6.21 Classification of Structured Objects

In this section, we present an algorithm which allows to answer the question whether the behavior of structured objects observed in a sequence of time windows belongs to a given temporal concept defined for structured objects.

Stratifying classifier μ_C^E constructed for table $\mathbf{G}_{\bowtie}^C$ enables the classification of structured objects to concept C. However, it may be applied for the tested object which is the arrangement of clusters of temporal paths of the behavioral graph for complex objects being parts of the examined structured object and clusters of temporal paths of the behavioral graph describing relation changes between the parts of the examined structured object. That is why it is necessary to construct earlier a suitable object (like in the table $\mathbf{G}_{\bowtie}^C$) in order to test of structured object. We present the algorithm of structured object classification. However, we assume that during execution of the Algorithm 6.5 the following elements are available:

- a test c-temporal information system $\mathbf{T}_{TS-P}$,
- a sweeping algorithm SA around objects from systems $\mathbf{T}_{TS-P}$,
- a test c-temporal information system $\mathbf{T}_{TS-R}$ representing features of relations between parts of structured objects determined by the algorithm SA,
- a formula of constraints $\Phi \in GDL(\mathbf{G}_{\otimes})$,
- a concept C defined by experts in the set $U_{\bowtie}$, representing some feature of structured objects of a fixed type T, where any object of the type T consists of parts, that are objects of types $T_1, ..., T_k$,
- a temporal concept table $\mathbf{G}_{\bowtie}^C$ for the concept C, constructed using the formula of constraints Φ,
- a linearly ordered set $(H, \leq_H)$ such that H is a set of labels of concepts layers and $\leq_H$ is a relation of linear order on the set H,
- a stratifying classifier μ_C^H constructed for the concept C on the basis the table $\mathbf{G}_{\bowtie}^C$.

The Algorithm 6.11 works as follows. On the input of the algorithm there is a description given of the behavior of a certain structured object, in the form of sequences of time windows: $S_1, ..., S_k, S_{k+1}$ which describe the behavior of a part of this object (sequences: $S_1, ..., S_k$) and the relation changes between the parts of this object (the sequence S_{k+1}). Next, on the basis of the sequences of time windows obtained on the input and with the usage of algorithms $MakeTimePath$ and $GetClusterRow$, there is created a new object u_{row} which has the structure of objects from the $\mathbf{G}_{\otimes}$ table. Next, it is checked whether object u_{row} satisfies the constraints expressed by formula Φ. If it happens this way, then object u_{row} is classified by the stratifying classifier μ_C^H, otherwise the algorithm returns the information that it cannot classify the behavior of the complex object whose description is given on the input of the algorithm.

Assuming that the operation of examining the satisfiability of formula Φ and the operation of classification with the help of classifier μ_C^H may be performed in constant time and considering the complexity of procedures $MakeTimePath$ and $GetClusterRow$, the pessimistic time complexity of the Algorithm 6.11 is of

Algorithm 6.11. Structured object classification to the temporal concept

```
Input: Family of sequences of time windows $S_1, ..., S_k, S_{k+1}$ describing
       behavior of a given structured object, where $S_i \in SEQ(\mathbf{T}_{TS-P})$,
       for $i = 1, ..., k$ and $S_{k+1} \in SEQ(\mathbf{T}_{TS-R})$
Output: Information about the membership of the observed structured
        object to the concept $C$
1  begin
2      Create empty list $row$
3      for $i := 1$ to $k + 1$ do
4          $path := MakeTimePath(S_i)$ (see Algorithm 6.8)
5          $pRow := GetClusterRow(path)$ (see Algorithm 6.10)
6          Add $pRow$ to the end of the list $row$
7      end
8      Create new object $u_{row}$ of the system $\mathbf{G}_{\otimes}$ on the basis values of
       attributes from the list $row$.
9      if $u_{row} \not\models_{GDL(\mathbf{G}_{\otimes})} \Phi$ then
10         return "Has nothing to do with"
11     else
12         return $\mu_C^H(u_{row})$
13     end
14 end
```

order $O(k \cdot s \cdot m + k \cdot n_{pf} + k \cdot n_c + k \cdot n_{cf})$, where k is the number of parts of which the examined structured objects consist, s is the length of each $S_1, ..., S_k, S_{k+1}$ sequences, m is the maximum number of nodes from the behavior graphs of individual parts of structured objects, n_{pf} is the maximum number of features used to define path properties in behavior graphs, n_c is the maximum number of path clusters in behavior graphs and n_{cf} is the maximum number of the applied features of such clusters.

6.22 Behavioral Graphs for Structured Objects

Analogously to the case of temporal concepts for unstructured complex objects, temporal concepts defined for structured objects may also be treated as nodes of a certain directed graph that we call *a behavioral graph for a structured object.* One can say, that the behavioral graph for a structured object expresses temporal dependencies on a higher level of generalization than the behavioral graph on lower level, i.e., the behavioral graph for unstructured objects.

Definition 53 (*A behavioral graph for a structured object*)**.**

1. *A behavioral graph for a structured object $\mathcal{G}$ is an ordered pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a nonempty and finite set of nodes (temporal concepts for structured objects) and $\mathcal{E}$ is a set of directed edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ (connections represent temporal dependencies between temporal concepts for structured objects).*

2. *A temporal path in the behavioral graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a sequence of nodes $v_1, ..., v_l$ such that for any $i \in \{1, ..., l-1\}$ an edge $(v_i, v_{i+1}) \in \mathcal{E}$. A number l is called a length of temporal path $v_1, ..., v_l$.*
3. *A family of all temporal paths with length l ($l > 0$) in the behavioral graph $\mathcal{G}$ is denoted by $PATH(\mathcal{G}, l)$.*

Bellow we present an example of behavioral graph for a structured object.

Example 32. In Fig. 33, we present exemplary behavioral graph for a structured object, that is a group of two vehicles: vehicle A and vehicle B, related to the standard overtaking maneuver. There are 6 nodes in this graph representing the following temporal concepts: *vehicle A is driving behind B on the right lane*, *vehicle A is changing lanes from right to left*, *vehicle A is moving back to the right lane*, *vehicle A is passing B when A is on the left lane and B is on the right lane*, *vehicle A is changing lanes from left to right and vehicle A is before B on the right lane.* There are 7 connections representing spatio-temporal dependencies between temporal concepts from nodes. For example, after the node *Vehicle A is driving behind B on the right lane* the behavior of these two vehicles matches to the node (temporal concept) *Vehicle A is changing lanes from right to left and B is driving on the right lane.* □

6.23 Behavioral Patterns

Both the behavioral graph for an unstructured object and the behavioral graph for a structured object may be used as a complex classifier enabling the identification of the behavioral pattern described by this graph. It is possible based on the observation of the behavior of a unstructured object or a structured object over a longer period of time and testing if the behavior matches the chosen path of the behavioral graph. If this is the case, it is stated that the behavior matches

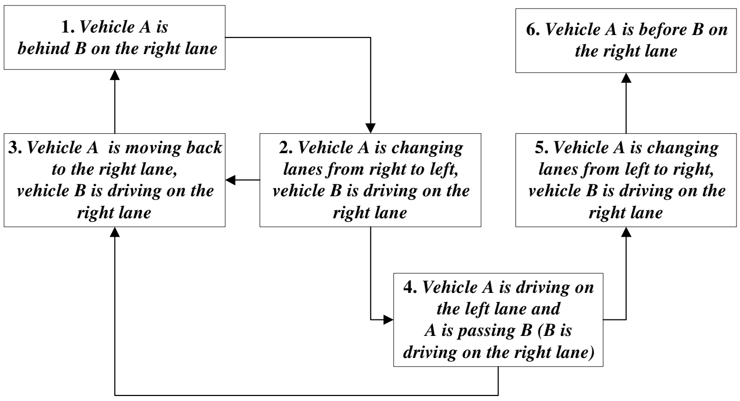

1. Vehicle A is behind B on the right lane
2. Vehicle A is changing lanes from right to left, vehicle B is driving on the right lane
3. Vehicle A is moving back to the right lane, vehicle B is driving on the right lane
4. Vehicle A is driving on the left lane and A is passing B (B is driving on the right lane)
5. Vehicle A is changing lanes from left to right, vehicle B is driving on the right lane
6. Vehicle A is before B on the right lane

Fig. 33. A behavioral graph for the maneuver of overtaking

Algorithm 6.12. Behavioral pattern identification

```
Input: Family of sequences of time windows $S_1, ..., S_k, S_{k+1}$ describing
       behavior of a given structured object, where:
  – $S_i \in SEQ(\mathbf{T}_{TS-P})$, for $i = 1, ..., k$ and $S_{k+1} \in SEQ(\mathbf{T}_{TS-R})$,
  – $Length(S_i) = z \cdot s$, where $z$ is a natural number and $s$ is a length of
    sequences of time windows used for identification of temporal concepts
    for structured objects, for $i = 1, ..., k+1$.

Output: The binary information ($YES$ or $NO$) about matching the
        observed structured object to the behavioral pattern
        represented by the graph $\mathcal{G}$
begin
    Create empty list $path$
    for $i := 0$ to $z - 1$ do
        $SS_1 := Subsequence(S_1, i \cdot s + 1, (i+1) \cdot s)$
        ...
        $SS_k := Subsequence(S_k, i \cdot s + 1, (i+1) \cdot s)$
        $SS_{k+1} := Subsequence(S_{k+1}, i \cdot s + 1, (i+1) \cdot s)$
        $v_{max} := v_1$
        $layer_{max} := \mu^H_{v_1}(SS_1, ..., SS_k, SS_{k+1})$
        for $j := 2$ to $m$ do
            $layer := \mu^H_{v_j}(SS_1, ..., SS_k, SS_{k+1})$
            if $layer_{max} \leq_H layer$ then
                $layer_{max} := layer$
            end
        end
        Add $v_{max}$ to the end of the list $path$.
    end
    if $path \in PATH(\mathcal{G}, z)$ then
        return $YES$
    else
        return $NO$
    end
end
```

behavioral pattern represented by this graph which enables detecting particular behaviors in a complex dynamical system.

In result, we can use a behavioral graph as a complex classifier for perception of the complex behavior of unstructured or structured objects. For this reason, a behavioral graph constructed for some complex behavior is called a *behavioral pattern*.

We present a detailed algorithm of behavioral pattern identification (see Algorithm 6.12). However, we assume that during execution of the Algorithm 6.12 the following elements are available:

– a test c-temporal information system $\mathbf{T}_{TS-P}$,
– a sweeping algorithm SA around objects from system $\mathbf{T}_{TS-P}$,
– a test c-temporal information system $\mathbf{T}_{TS-R}$ representing features of relations between parts of structured objects determined by the algorithm SA,
– a behavioral graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of structured objects of a fixed type T determined by the algorithm SA such that $\mathcal{V} = \{v_1, ..., v_m\}$,
– a linearly ordered set $(H, \leq_H)$ such that H is a set of labels of concepts layers and $\leq_H$ is a relation of linear order on the set H,
– a family of stratifying classifiers $\mu^H_{v_1}, ..., \mu^H_{v_m}$ constructed for concepts corresponding to nodes $v_1, ..., v_m \in \mathcal{V}$.

The way of working Algorithm 6.12 is the following. On the input there is a description given of the behavior of a certain structured object, in the form of sequences of time windows: $S_1, ..., S_k, S_{k+1}$ which describe the behavior of a part of this object (sequences: $S_1, ..., S_k$) and the relation changes between the parts of this object (sequence S_{k+1}). The length of these sequences is established and equals $z \cdot s$, where z is a fixed natural number and s is the length of sequences of time windows needed to identify a single temporal concept for structured objects. Further, for the all subsequence families of the length s isolated from the input sequences, stratifying classifiers are applied constructed for concepts corresponding to all nodes of behavior graph $\mathcal{G}$. Thus, it is possible to chose such a node for each subsequence family that the classifier corresponding to it classifies the subsequence family to the possibly lowest layer (if there are more such nodes the choice among them is nondeterministic). The chosen node is put at the end of the path represented using the *path* list. After choosing the node of graph $\mathcal{G}$ for all subsequence families and what follows completing the *path* list, the return of decision value occurs which tells us whether the examined structured object matches the behavioral pattern represented by graph $\mathcal{G}$ or not. Namely, the YES decision is returned when the list *path* is a path in graph $\mathcal{G}$, otherwise the NO value is returned.

The above algorithm is able to classify a given structured object only when all subsequence families isolated from the input family of the sequence of time windows may be classified. However, this is possible only when each subsequence family is classified at least by one of the stratifying classifiers available for nodes from set $\mathcal{V}$.

Let us notice that in terms of computational complexity the time cost of executing the Algorithm 6.12 is equal, with an accuracy of constant, the sum of costs of $(z \cdot m)$-times execution of Algorithm 6.11, where z is the number of the subsequences of the length s into which the input sequences are divided and $m = card(\mathcal{V})$.

Example 33. Let us study the behavioral graph presented in Fig. 33 for a group of two objects-vehicles (vehicle A and vehicle B) related to the standard overtaking maneuver. We can see that the path of temporal concepts with indexes "1, 2, 3, 1, 2, 4" matches a path from this behavioral graph, while the path with indexes: "6, 5, 4" doesn't match any path from this behavioral graph (this path can match some other behavioral patterns). $\square$

A path of temporal patterns (that makes it possible to identify behavioral patterns) should have a suitable length. In the case where the length is too short, it may be impossible to discern one behavioral pattern from another pattern. For example, we can make a mistake between an overtaking and a passing by a vehicle in traffic.

6.24 Discovering Rules for Fast Elimination of Behavioral Patterns

In this section, we present how the method of behavioral patterns identification presented in Section 6.23 can be speeded up using special decision rules. Let us assume that we have a family of behavioral patterns $BP = \{b_1, ..., b_n\}$ defined for structured objects (or parts of a given structured object). For any pattern b_i from the family BP one can construct a complex classifier based on a suitable behavioral graph (see Section 6.23) that makes it possible to answer the question: "Does the behavior of the investigated structured object (or the part of a structured object) match the pattern b_i?". The identification of behavioral patterns of any structured object is performed by investigation of a sequence of time windows registered for this object during some period (sometimes quite long). This registration of time windows is necessary if we want to avoid mistakes in identification of the investigated structured object. However, in many applications, we are forced to make a faster (often in real-time) decision if some structured object matches the given behavioral pattern. In other words, we would like to check the investigated structured object at once, that is, using the first or second time window of our observation only. This is very important from the computational complexity point of view, because if we investigate complex dynamical systems, we usually have to investigate many structured objects. Hence, faster verification of structured objects can help us to optimize the process of searching among structured objects matching the given behavioral pattern.

The verification of complex dynamical systems consisting of some structured objects can be speeded up by using some special decision rules, that are computed by an Algorithm 6.13 (see also Fig. 34).

Any decision rule computed by the Algorithm 6.13 expresses a dependency between a temporal concept and the set of behavioral patterns that are not matching this temporal concept. Such rules can make it possible to exclude very quickly many parts of a given complex object as irrelevant for identification of a given behavioral pattern. This is possible because these rules can be often applied at once, that is after only one time window of our observation. Let us consider a very simple illustrative example. Assume we are interested in the recognition of overtaking that can be understood as a behavioral pattern, defined for the group of two vehicles. Using the methodology presented above, we can obtain the following decision rule:

– `If the vehicle A is overtaking B then the vehicle B is driving on the right lane.`

Algorithm 6.13. Discovering decision rules for fast elimination of behavioral patterns

Input:
- c-temporal information system $\mathbf{T}$,
- a family of behavioral patterns $BP = \{b_1, ..., b_n\}$ defined for structured objects or parts of structured objects.

Output: Decision rules for fast elimination of behavioral patterns from the family BP

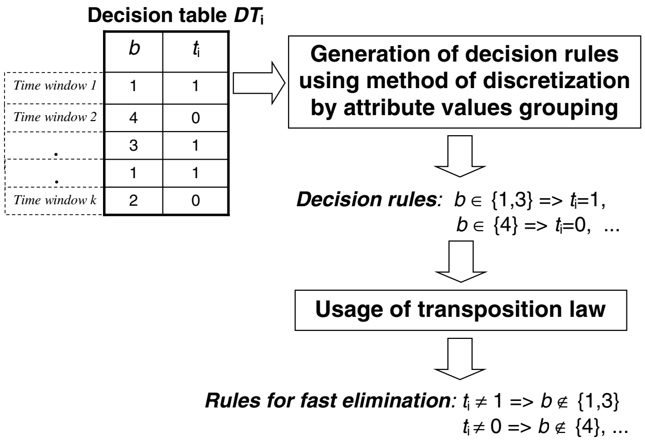
1. Define a family of temporal concepts $TC = \{t_1, ..., t_m\}$ that have influence on matching investigated structured objects to behavioral patterns from family BP (defined on the basis of information from time windows from the set $TW(\mathbf{T})$).
2. Construct classifiers for all temporal concepts from TC using the method from Section 6.9.
3. For any temporal concept t_i from the family TC create a decision table DT_i, that has the following structure:
   (a) any row of the table DT_i is constructed on the basis of information registered during a period that is typical for the given temporal concept t_i,
   (b) the condition attribute b of table DT_i registers the index of behavioral pattern from the family BP (the index computation is based on observation that any complex classifier from BP can check for the investigated structured objects if there is a sequence of time windows matching the given behavioral pattern and starting from a given time window),
   (c) the decision attribute of the table DT_i is computed on the basis of values returned by classifier constructed for t_i in previous step.
4. Compute decision rules for DT_i using methods of discretization by attribute values grouping (see Section 2.2).

After applying the transposition law, we obtain the following rule:

– `If the vehicle B is not driving on the right lane then the vehicle A is not overtaking B.`

The last rule (see also Fig. 35) allowing fast verification whether the investigated structured object (two vehicles: A and B) is matching the behavioral pattern of overtaking.

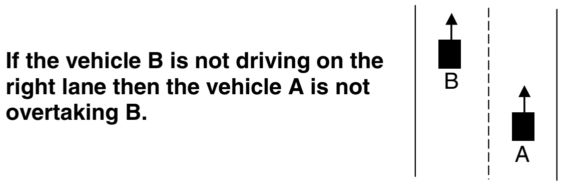
Of course, in case of the considered complex dynamical system, there are many other rules that can help us in the fast verification of structured objects related to the overtaking behavioral pattern. Besides, there are many other behavioral patterns in this complex dynamical system and we have to calculate rules for them using the methodology presented above.



Decision table DT_i
b
t_i
Time window 1
Time window 2
Time window k
Generation of decision rules using method of discretization by attribute values grouping
Decision rules: $b \in \{1,3\} \Rightarrow t_i=1$, $b \in \{4\} \Rightarrow t_i=0$, ...
Usage of transposition law
Rules for fast elimination: $t_i \neq 1 \Rightarrow b \notin \{1,3\}$, $t_i \neq 0 \Rightarrow b \notin \{4\}$, ...

	b	t_i
Time window 1	1	1
Time window 2	4	0
.	3	1
.	1	1
Time window k	2	0

Fig. 34. The scheme of rules extraction for the fast elimination of behavioral patterns from data tables



If the vehicle B is not driving on the right lane then the vehicle A is not overtaking B.
B
A

Fig. 35. The illustration of the decision rule for fast elimination of behavioral pattern

The presented method, that we call *the method for on-line elimination of non-relevant for a given behavioral pattern parts of complex object* (ENP method [177, 178]), is not a method for behavioral pattern identification. However, this method allows to eliminate some paths of a given complex object behavior that are not relevant for checking if this object matches a given behavioral pattern. After such elimination the complex classifiers based on a suitable behavioral graphs should be applied to the remaining complex objects.

6.25 Experiments with Road Simulator Data

To verify the effectiveness of classifiers based on behavioral patterns, we have implemented the algorithms from the *Behavioral Patterns* library (BP-lib), which is an extension of the RSES-lib 2.1 library forming the computational kernel of the RSES system (see [15]). Our experiments have been performed on the data sets obtained from the road simulator (see Appendix A) and on the medical data sets. In this section we report results of experiments preformed on the data sets obtained from the road simulator. Results obtained for the medical data sets are presented in Section 6.26.

In the case of experiments on the data sets obtained from the road simulator, we have applied the *train-and-test* method. A training set consisted of about 17 thousands objects generated by the road simulator during one thousand of simulation steps. Whereas, a testing set consisted also of about 17 thousands objects collected during another (completely different) session with the road simulator.

In our experiments, we compared the quality of three classifiers: *rough set classifier with decomposition* (RS-D), *Behavioral Pattern classifier* (BP) and *Behavioral Pattern classifier with the fast elimination of behavioral patterns* (BP-ENP).

For induction of RS-D, we employed RSES system generating the set of minimal decision rules by algorithm LEM2 (see Section 2.4) that are next used for classification of situations from the testing data. However, we had to use the method of generating decision rules joined with a standard decomposition algorithm from the RSES system (see Section 2.7). This was necessary because the size of the training table was too large for the direct generation of decision rules. The classifiers BP is based on behavioral patterns (see Section 6.23), whilst BP-ENP are based on behavioral patterns too but with application of fast elimination of behavioral patterns related to the investigated group of objects (see Section 6.24).

In application of BP and BP-ENP methods the distance between time points was constant, that is time points were recorded after each stage of the simulation. The prediction of temporal concepts for individual vehicles was performed on the basis of time windows whose duration equals 3 time points, whereas the prediction of temporal concepts for pairs of vehicles was performed on the basis of sequence of time windows whose duration was equal 2. Finally, the behavioral pattern was recognized on the basis of vehicle observation over 2 sequences of time windows, that is on the basis of vehicle observation over 4 time windows. The tested object for the analyzed behavioral pattern was, thus, the sequence of 9 successive time points. Therefore, classifier RS-D used the table whose objects were sequences of 9 successive time points, whereas the attributes gave the values of sensor attributes for all these points. The value of decision attribute was given by the expert in the same manner for all three classification methods.

We compared RS-D, BP, and BP-ENP classifiers using the accuracy, the coverage, the real accuracy, the accuracy for positive examples (the sensitivity or the true positive rate), the coverage for positive examples, the real accuracy for positive examples, the accuracy for negative examples (the specificity or the true negative rate), the coverage for negative examples and the real accuracy for negative examples (see Section 2.9).

In order to determine the standard deviation of the obtained results each experiment was repeated for 10 pairs of tables (training table + testing table). Therefore, 20 tables in total were applied (collected during 20 completely different sessions with the road simulator).

Table 8 shows the results of applying these classification algorithms for the concept related to the *overtaking* behavioral pattern.

Table 8. Results of experiments for the overtaking pattern

Decision class	Method	Accuracy	Coverage	Real accuracy
Yes (overtaking)	RS-D	0.875 ± 0.050	0.760 ± 0.069	0.665 ± 0.038
	BP	0.954 ± 0.033	1.000 ± 0.000	0.954 ± 0.033
	BP-ENP	0.947 ± 0.031	1.000 ± 0.000	0.947 ± 0.031
No (no overtaking)	RS-D	0.996 ± 0.001	0.989 ± 0.003	0.985 ± 0.001
	BP	0.990 ± 0.004	1.000 ± 0.000	0.990 ± 0.004
	BP-ENP	0.990 ± 0.004	1.000 ± 0.000	0.990 ± 0.004
All classes (Yes + No)	RS-D	0.993 ± 0.002	0.980 ± 0.004	0.973 ± 0.002
	BP	0.988 ± 0.003	1.000 ± 0.000	0.988 ± 0.003
	BP-ENP	0.988 ± 0.003	1.000 ± 0.000	0.988 ± 0.003

One can see that in the case of perception of the overtaking maneuver (decision class Yes) the accuracy and the real accuracy of algorithm BP are higher than the accuracy and the real accuracy of algorithm RS-D for the analyzed data set. Besides, we see that the accuracy of algorithm BP-ENP (for decision class YES) is only 0.7 percent lower than the accuracy of algorithm BP. Whereas, the algorithm BP-ENP allows us to reduce the time of perception, because during perception we can usually identify the lack of overtaking earlier than in the algorithm BP. This means that it is not necessary to collect and investigate the whole sequence of time windows (that is required in the BP method) but only some first part of this sequence. In our experiments with the classifier BP-ENP it was enough to use on average $59.7\% \pm 1.5\%$ percent of the whole time window sequence for objects from the decision class No (the lack of overtaking in the sequence of time windows). However, it should be stressed that this result concerns only identification of vehicle groups which were preliminarily selected by the sweeping algorithm (see Section 6.13), whereas in comparison with the number of time windows needed to analyze all possible two-element vehicle groups, in using the sweeping algorithm and BP-ENP method the number of analyzed time windows constitutes only $2.3\% \pm 0.1\%$ of the number of time windows needed to analyze all 2-element vehicle groups.

6.26 Risk Pattern Identification in Medical Data: Case Study

An identification of some behavioral patterns can be very important for identification or prediction of behavior of a complex dynamical system, especially when behavioral patterns describe some dangerous situations. In this case, we call such behavioral patterns *risk patterns* and we need some tools for their identification. If in the current situation some risk patterns are identified, then the control object (a driver of the vehicle, a medical doctor, a pilot of the aircraft, etc.) can use this information to adjust selected parameters to obtain the desirable behavior of the complex dynamical system. This can make it possible to overcome inconvenient or unsafe situations. For example, a very important element of the treatment of the infants with respiratory failure is the appropriate assessment of the risk of death. The appropriate assessment of this risk leads to the decision of particular method and level of treatment. Therefore, if some complex behavior of an infant that causes a danger of death is identified, we can try to change her/his behavior by using some other methods of treatment (may be more radical) in order to avoid the infant's death. In this section we describe how the presented approach in previous sections can be applied to identification of the infants' death risk caused by respiratory failure (see Appendix B). In this approach, a given patient is treated as an investigated complex dynamical system, whilst diseases of this patient (RDS, PDA, sepsis, Ureaplasma and respiratory failure) are treated as complex objects changing and interacting over time.

It is also worthwhile mentioning that the research reported in this section is a continuation, in some sense, of the previous research on the survival analysis (see [324, 325, 326]).

Medical Temporal Patterns. As we wrote before (see, e.g, Section 6.4), the concepts concerning the properties of complex objects at the current time point in connection with the previous time point are a way of representing very simple behaviors of the complex objects. These concepts, that we call *elementary concepts*, usually characterize a status of sensor's values. In the case of our medical example (the treatment of the infants with respiratory failure), we can distinguish the following elementary concepts such as *low value of FiO2* (the percent concentration of oxygen in the gas entering the lungs), *increase in FiO2* , *decrease in PaO2* (the arterial oxygen tension), *decrease in PaO2/FiO2*, *low creatinine serum (blood) level.* However, a perception of more complex behaviors requires identification of elementary concepts over a longer period called a *time window* (see Section 6.6). Therefore, if we want to predict such more complex behaviors or discover a behavioral pattern, we have to investigate elementary concepts registered in the current time window. Such investigation can be expressed using temporal patterns. For example, in the case of the medical example one can consider patters expressed by following questions: "Did PaO2/FiO2 increase in the first point of the time window?", "Was PaO2/FiO2 stable in the time window?", "Did the PaO2/FiO2 increase before the closing of PDA?" or "Did the PaO2/FiO2 increase before a PaO2/FiO2 decrease occurred?". Notice that all such patterns ought to be defined by a human, medical expert using domain knowledge accumulated for the respiratory failure disease.

Behavioral Graph for a Disease. The temporal patterns can be treated as new features that can be used to approximate temporal concepts (see Section 6.6). In the case of the treatment of infants with respiratory failure one can define temporal concepts such as "Is the infant suffering from the RDS on level 1?", "Was an multi-organ failure detected?", or "Is the progress in multi-organ failure on level 4?".

Temporal concepts defined for objects from a complex dynamical system and approximated by classifiers, can be treated as nodes of a graph called a *behavioral graph* (see Section 6.11), where connections between nodes represent temporal dependencies. Fig. 36 presents a behavioral graph for a single patient exhibiting

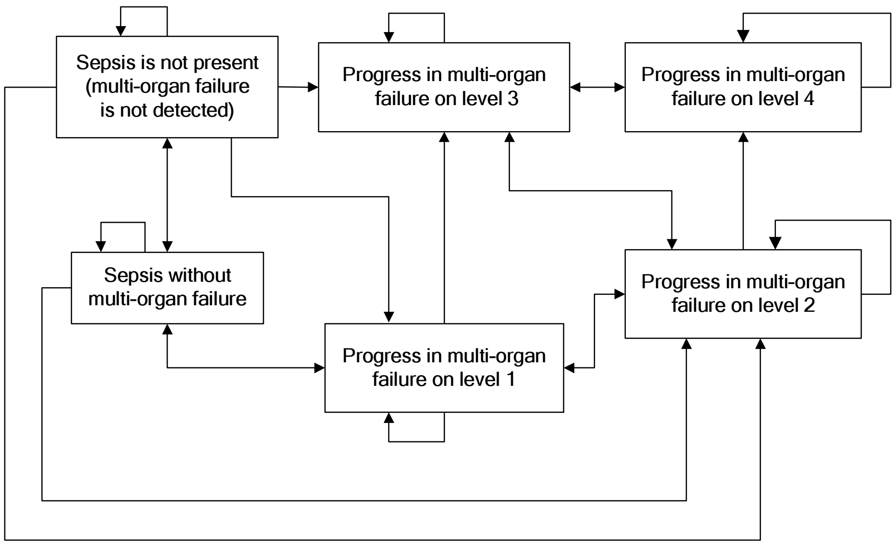

Sepsis is not present (multi-organ failure is not detected)
Progress in multi-organ failure on level 3
Progress in multi-organ failure on level 4
Sepsis without multi-organ failure
Progress in multi-organ failure on level 2
Progress in multi-organ failure on level 1

Fig. 36. A behavioral graph of sepsis by analyzing the multi-organ failure

a behavioral pattern of patient by analysis of the organ failure caused by sepsis. This graph has been created on the basis of observation of medical data sets (see Appendix B) and the SOFA scale (Sepsis-related Organ Failure Assessment) (see [327, 328] for more details).

In this behavioral graph, for example, connections between node "Progress in multi-organ failure on level 1" and node "Progress in multi-organ failure on level 3" indicates that after some period of progress in organ failure on level 1 (rather low progressing), a patient can change his behavior to the period, when progress in organ failure is high. In addition, a behavioral graph can be constructed for different kinds of diseases (like RDS, PDA, Ureaplasma) (see Appendix B) or groups of diseases represented for example by the respiratory failure (see Fig. 37).

Medical Risk Pattern. The temporal concepts defined for structured objects and approximated by classifiers, are nodes of a new graph, that we call *a behavioral graph for a structured object* (see Section 6.22). In Fig. 37, we present an exemplary behavioral graph for group of four diseases: sepsis, Ureaplasma, RDS and PDA, related to the behavior of the infant during high death risk period due to respiratory failure. This graph has been created on the basis of observation of medical data sets (see next subsection) and with support of medical experts. There are 16 nodes in this graph and 21 connections represented spatio-temporal dependencies between temporal concepts from nodes. For example, after the node "Stabile and mild respiratory failure in sepsis" the behavior of patient can match the node "Exacerbation of respiratory failure from mild to moderate in sepsis".

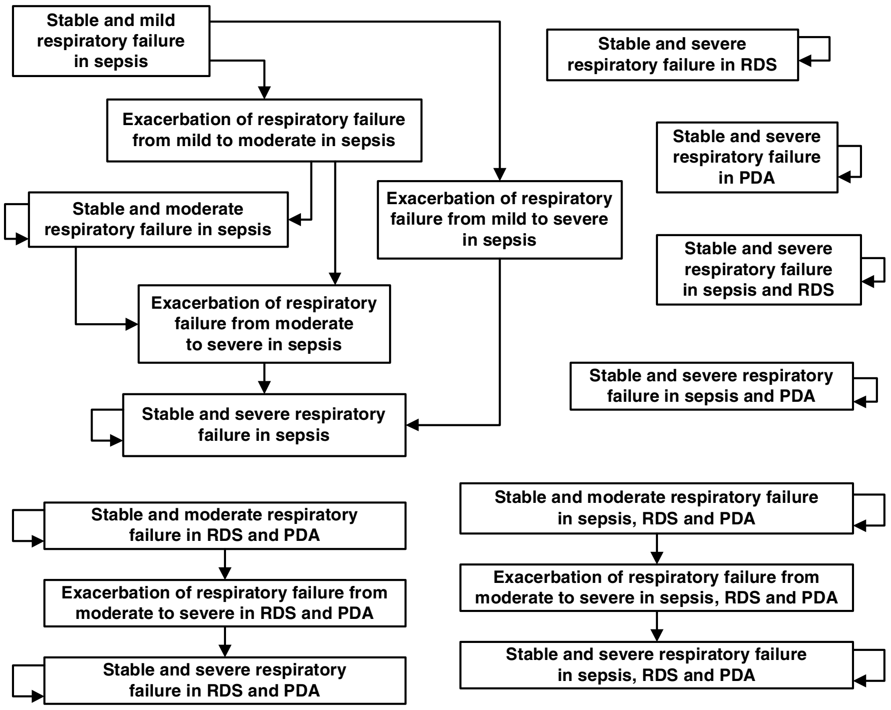

Stable and mild respiratory failure in sepsis
Exacerbation of respiratory failure from mild to moderate in sepsis
Stable and moderate respiratory failure in sepsis
Exacerbation of respiratory failure from mild to severe in sepsis
Exacerbation of respiratory failure from moderate to severe in sepsis
Stable and severe respiratory failure in sepsis
Stable and severe respiratory failure in RDS
Stable and severe respiratory failure in PDA
Stable and severe respiratory failure in sepsis and RDS
Stable and severe respiratory failure in sepsis and PDA
Stable and moderate respiratory failure in RDS and PDA
Exacerbation of respiratory failure from moderate to severe in RDS and PDA
Stable and severe respiratory failure in RDS and PDA
Stable and moderate respiratory failure in sepsis, RDS and PDA
Exacerbation of respiratory failure from moderate to severe in sepsis, RDS and PDA
Stable and severe respiratory failure in sepsis, RDS and PDA

Fig. 37. A behavioral graph of the infant during high death risk period due to respiratory failure

This behavioral graph is an example of risk pattern. We can see that the path of temporal patterns: ("Stable and mild respiratory failure in sepsis", "Exacerbation of respiratory failure from mild to severe in sepsis", "Stable and severe respiratory failure in sepsis") matches a path from this behavioral graph, while the path: ("Stable and severe respiratory failure in sepsis", "Exacerbation of respiratory failure from moderate to severe in sepsis", "Stable and moderate respiratory failure in sepsis") doesn't match any path from this behavioral graph.

Experiments with Medical Data. In this section we present results of experiments performed for obtained from Neonatal Intensive Care Unit, First Department of Pediatrics, Polish-American Institute of Pediatrics, Collegium Medicum, Jagiellonian University, Krakow, Poland. The data were collected between 2002 and 2004 using computer database NIS, i.e, Neonatal Information System (see [329]). The detailed information about treatment of 340 newborns are available in the data set, such as perinatal history, birth weight, gestational age, lab tests results, imagine techniques results, detailed diagnoses during hospitalization, procedures and medication were recorded for the each patient. The study group included prematurely born infants with the birth weight $\leq 1500g$, admitted to the hospital before end of the 2 day of life. Additionally, the children suffering from the respiratory failure but without diagnosis of RDS, PDA, sepsis or Ureaplasma infection during their entire clinical course were excluded from the study group (193 patients stayed after the reduction).

In our experiments, we used one data table extracted from the NIS system, that consists of 11099 objects. Each object of this table describes parameters of one patient in single time point.

The aim of conducted experiments was to check the effectiveness of the algorithms described in this paper in order to predict the behavioral pattern related to a high risk of death of infants. This pattern was defined by experts (see Fig. 37). It is worth adding that as many as 90.9% of infants whose behavior matched this pattern died shortly after (this fact results from a simple analysis of medical data set which were gathered).

As a measure of classification success (or failure) we use: the accuracy, the coverage, the real accuracy, the accuracy for positive examples (the high risk of death), the coverage for positive examples, the real accuracy for positive examples, the accuracy for negative examples (the low risk of death), the coverage for negative examples and the real accuracy for negative examples (see Section 2.9).

We have applied the *train-and-test* method. However, because of the specificity of the analyzed data the method of data division differed slightly from the standard method. Namely, in each experiment the whole patient set was randomly divided into two groups (training and testing one). The same number of patients belonged to each of these groups, at the same time patients who died and those who survived were divided separately. In other words, in each of two groups the number of dead patients and those who survived was the same. This division of data was necessary because the correlation between patient's death and the fact of matching patient's behavior the considered behavioral pattern is very strong. Obviously, the information about whether the patient died or survived the treatment was not available during learning and testing process of classifiers.

In the discussed experiments the distance between time points recorded for a specific patient was variable, that is, various time points of different frequencies were recorded in the data over different periods of time. For instance, if patient's condition was serious, then quite often (*e.g.*, every two hours) parameters representing his or her condition were registered and recorded for this patient, whereas if the patient's condition was good and stable, then the information about this patient was recorded rather rarely (*e.g.*, once a day). In relation to this, although the prediction of temporal concepts for individual disease (RDS, PDA, sepsis, Ureaplasma) was always performed on the basis of time windows having 2 time points, then in practice these windows had very different temporal durations. This way, the duration of time windows was in a certain way determined by experts. However, the prediction of temporal concepts for respiratory failure was performed on the basis of the sequence of time windows whose duration was equal 2. Finally, the pattern of high death risk was recognized on the basis of patient observation for 3 sequences of time windows that is on the basis

Table 9. Results of experiments for the risk pattern of death due to respiratory failure

Decision class	Accuracy	Coverage	Real accuracy
Yes (the high risk of death)	0.994 ± 0.008	1.0 ± 0.000	0.994 ± 0.008
No (the low risk of death)	0.938 ± 0.012	1.0 ± 0.000	0.938 ± 0.012
All classes (Yes + No)	0.958 ± 0.010	1.0 ± 0.000	0.958 ± 0.010

of observation over 6 time windows. A tested object for the analyzed behavioral pattern was, therefore, the sequence of 7 successive time points.

As a result of the above mentioned division of patients into training and testing ones, each of these parts made it possible to create approximately 6000 time windows having duration of 7 time points. Time windows created on the basis of training patients created a training table for a given experiment, while time windows created on the basis of tested patients created a test table for the experiment.

In order to determine the standard deviation of the obtained results each experiment was repeated for 10 random divisions of the whole data set.

Table 9 shows the results of applying this algorithm for the concept related to the risk pattern of death due to respiratory failure. Together with the results we present a standard deviation of the obtained results.

Notice, that the accuracy of decision class Yes in medical statistics (see [260] and Section 2.9) is called a *sensitivity* (the proportion of those cases having a true positive test result of all positive cases tested), whereas the accuracy of decision class No is called a *specificity* (the proportion of true negatives of all the negative samples tested). We see both main parameters of our classifier (i.e., sensitivity and specificity) are sufficiently high.

Experimental results showed that the suggested method of behavioral patterns identification gives good results, also in the opinion of medical experts (compatible enough with the medical experience) and may be applied in medical practice as a supporting tool for infants suffering from respiratory failure monitoring.

Some results of our experiments on medical data were surprising even for medical experts (*e.g.*, very low frequency of fatal cases in infants with Ureaplasma infection). Therefore, one can say that our tools were useful for development of new interesting observation and experience.

Finally, let us notice that the specific feature of the methods considered here is not only high accuracy (with low standard deviation) but also very high coverage (equal 1.0).

7 Automated Planning Based on Data Sets and Domain Knowledge

Behavioral patterns described in Section 6 may be very useful for effective complex dynamical systems monitoring, particularly when certain behavioral patterns are connected to undesirable behaviors of complex objects. If during the observation of complex dynamical system such a pattern is identified then the control module may try to change, using appropriate actions, the behavior of the system in such a way as to get the system out of an uncomfortable or dangerous situation. However, these types of short-term interventions may not be sufficient for a permanent rescuing the system out of an undesirable situation. Therefore, very often the possibility of using some methods of automated planning is considered.

Automated planning is a branch of artificial intelligence that concerns the realization of strategies or action sequences (called as plans), typically for execution by intelligent agents, autonomous robots and unmanned vehicles, that can change their environment (see, *e.g.*, [70, 76, 273]). The essential inputs for planning are an initial world state, a repertoire of action for changing that world, and a set of goals. The purpose of plan is to achieve one or more goals. The form of the plan is commonly just a linear sequence of actions or acyclic directed graph of actions.

In the case of the control of complex dynamical systems, automated generated plans can be used to carry out a given complex object to more comfortable or safer situation (see, *e.g.*, [70, 72, 74, 76, 180, 181]). Such plans may be performed by (or on) unstructured objects or structured objects, i.e., by each part of any structured object separately with a presence of complete synchronization of actions performed by (or on) individual parts of the structured object).

7.1 Classical Algorithms of Automated Planning

Classical planning algorithms can be classified according to how they structure the search space. There are three very common such classes, like: *state space planners*, *plan space planners* and *planners encoding the planning problem as a problem of some other kind* (see, *e.g.*, [70, 71, 74, 330, 331]).

State Space Planners. In the first class of classical planning algorithm are very early planners, *e.g.*, STRIPS (see [332]), and some successful recent planners, like Graphplan (see [333]). These algorithms are based on searching in the state space, where such searching is most often done either by forward-chaining, i.e., searching from the initial state to the goal state, or by backward-chaining, i.e., searching from the goal state to the initial state (see, *e.g.*, [70, 71, 74] for more details).

Plan Space Search. In the second class of planning algorithms are causal link planners and constraint-based planners (see, *e.g.*, [74]). In this case, the plan space consists of incomplete plans, which, in contrast to the notion of plan in the state space view, do not have to be sequences or actions or sets of parallel actions. One alternative is to view the plan as a partially ordered set of actions. The potential advantage of this view is that one partially ordered set can represent many linear plans, and that the planner needs only to enforce the orderings that are absolutely necessary, whereas in a linear plan, many action orderings are quite arbitrary (see, *e.g.*, [70, 71, 74] for more details).

Encoding Planning as a Different Problem. A third class of classical planners encode the planning problem as a problem of some other kind and solve this problem. This approach was first used in SATPLAN (see, e.g, [71, 334, 335, 336, 337]) which converts the planning problem instance into an instance of the Boolean satisfiability problem, which is then solved using a method for establishing satisfiability such as the DPLL algorithm or WalkSAT (see, *e.g.*, [338]). Other methods of planning in which the planning problem has been encoded as a problem of some other kind are methods based on a linear programming (see, *e.g.*, [339, 340]) or using constraint satisfaction problems (CSP) (see, *e.g.*, [341]).

7.2 Domain Dependent Automated Planning

Planning applications in practice may be large and involve complicated actions, but they commonly also have a great deal of structure, known to people experienced with the domain. There are many potential plans that are (to the human domain expert) obviously useless, and sometimes simple criteria can be found for sifting out the "promising" partial solutions. If this knowledge is encoded and given to the planner, it should help to speed up the process of finding a plan. This idea leads to what is called "domain-dependent planning".

There are many planning methods which use domain knowledge. At this section we briefly discuss a few most widely known exemplary approaches, that is *the planning with learning*, *the planning with time*, *the planning with incomplete information*, *the hierarchical task network planning* and *the domain-dependent search control.*

Learning in Planning. Generally speaking, machine learning techniques can be used to extract useful knowledge, from solutions to similar problem instances in the past or from previous failed attempts to solve the present problem instance. This has been used to improve planning efficiency and to improve plan quality (see, *e.g.*, [70, 71] for more details). For instance, two of the earliest systems to integrate machine learning and planning are SOAR (see, *e.g.*, [35, 302]), a general cognitive architecture for developing systems that exhibit intelligent behaviour, and PRODIGY, an architecture that integrates planning and learning in its several modules (see, *e.g.*, [303, 342]).

The approach to automated planning presented in this paper may also be included into the approaches integrating methods of machine learning with classical planning. However, there are significant differences between methods known from literature and methods presented in this paper (see Section 7.3 for more details).

Planning with Time. In classical planning actions are assumed to take "unit time". This assumption is not critically important, as long as there is no deadline to meet and one does not try to optimize the actual execution time. An early planner to deal with actions of different duration and goals with deadlines is Deviser (see [343]). It is based on the idea of partial-order planning, in which the simple partial order over the actions in the plan are replaced by more complex constraints on their starting and ending times. The idea has been picked up in several later planners (see, *e.g.*, [70, 71, 74] for more details). Recently, temporal planning has become a very active area of research, and almost every classical planning approach has been adapted to deal with durative actions.

Planning with Incomplete Information. Another assumption made by most planners is that all relevant information is available in the problem description, and that the effects of actions are perfectly predictable. There have been several approaches to relaxing this assumption, by introducing probabilistic information. The probabilistic information has been used with state space planners, partial-order planners but most of all in the form of Markov Decision Problems (see, *e.g.*, [70, 71, 74] for more details).

Hierarchical Task Network Planning. Hierarchical Task Network (HTN) planning (see, *e.g.*, [70, 344, 345]) is like classical planning in that each state of world is represented by set of atoms, and each action corresponds to deterministic state transition. However, HTN planners differ from classical planners in what they plan for and how they plan for it. In an HTN planner, the objective is not to achieve a set of goals but instead to perform some set of tasks. The input to the planning system includes a set of operators similar to those of classical planning and also a set of methods. Each of which is a prescription for how to decompose some task into some set of subtasks (smaller tasks). Planning proceeds by decomposition non-primitive tasks recursively into smaller and smaller subtasks, until primitive tasks are reached that can be performed directly using the planning operators (see [70, 71] for more details).

Domain-Dependent Search Control. Many search-based planners allow the speciation of domain-dependent heuristics and rules for controlling and reducing search. For example, two recent planners, TLPlan (see [346]) and TALplanner (see [347, 348]) depend entirely on domain-specific search control, given in the form of logical formulas.

7.3 Automated Planning for Complex Objects

In the all aforementioned approaches to automated planning for complex objects, it is assumed that we know the current state of the complex object, which results from a simple analysis of current values of this object's available parameters. In other words the state of the complex object may be directly read from the values of its parameters or from a simple analysis of dependencies between these values. For example, if we consider a classic *blocks world problem* (see, *e.g.*, [74, 349]) which consists in planning the way of arranging available blocks on the table to make a determined construction, the state of the object is the information about the current placement of the blocks; and at the same time while planning the arrangement the answers to the three following questions are taken into account.

1. *Is a given block lying directly on the table?*
2. *Is another block lying on a given block?*
3. *Is another specific block lying on a given block?*

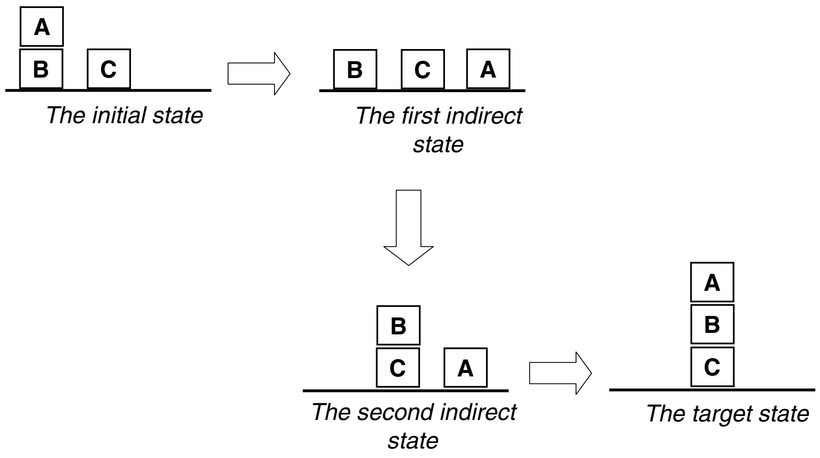


Fig. 38. Four states in the blocks world problem

For example, for the state in which there are three blocks available: A, B and C where blocks B and C are lying directly on the table and block A is lying on block B (initial state from Fig. 38), the description of such a state could be described in a natural language with the help of the five following facts:

1. *block A is lying directly on the table*,
2. *block B is lying directly on the table*,
3. *block C is lying on block A*,
4. *there is no block on block C*,
5. *there is no block on block B.*

Let us notice that in the above example concerning arranging a predetermined construction out of blocks, the description of the current state can be directly read from the information about the current values of their parameters, that is, from the information about the arrangement of blocks in relation to the table and other blocks. In the meantime, in complex dynamical system the state of the complex object is often expressed in a natural language using vague spatio-temporal conditions whose authenticity cannot be checked on the basis of a simple analysis of the available information on the object. For example, while planning treatment the condition of an infant who suffers from respiratory failure may be described by the following condition.

– *Patient with RDS type IV, persistent PDA and sepsis with mild internal organs involvement* (see Appendix B for mor medical details).

Stating the fact that a given patient is in the above condition requires the analysis of the examination result of this patient registered over a certain period of time with a great support of domain knowledge deriving from experts (medical doctors). Conditions of this type can be represented by complex concepts and the identification of the condition is a check if the analyzed objects belong to this concept or not. However, the identification of such states requires approximation concepts representing them with the help of classifiers using data sets and domain knowledge. In a few next sections we describe the automated planning method for unstructured complex objects whose states are described using this type of complex concepts.

7.4 Planning Rules and Planning Graphs

The basic concept used in this paper for automated planning is *a planning rule.* It is a simple tool for modeling changes of the states of complex objects as a result of applying (performing) actions.

Definition 54 (*A planning rule*)**.** *Let S be the set of complex objects' states of the fixed type T and set A be the set of actions whose application causes the complex objects to change state from one to another. Each expression of the form: $(s_l, a) \to s_{r_1}|s_{r_2} \ldots |s_{r_k}$, where $s_l, s_{r_1} \ldots s_{r_k} \in S$ and $a \in A$ is called a planning rule of complex object of type T. Moreover, expression (s_l, a) is called a predecessor of the planning rule and expression $s_{r_1}|s_{r_2} \ldots |s_{r_k}$ is called a successor of the planning rule.*

Such rule can be used to change the state s_l of a complex object, using the action a to some state from the right hand side of a rule. But the result of applying such a rule is nondeterministic, because there are usually many states on the right hand side of a planning rule.

Example 34. Let us consider the planning rule from Fig. 39. This is the planning rule for treating RDS (respiratory distress syndrome) obtained from domain knowledge (see Appendix B). The rule may be applied when RDS with very severe hypoxemia is present. The application of the rule consists in performing a medical action utilizing the respirator in the MAP3 mode (see Example 36 for more medical details). As an effect of the application of this action at the following time point of observation (*e.g.*, the following morning) the patient's condition may remain unchanged or improve so as to reach the condition of RDS with severe hypoxemia. □

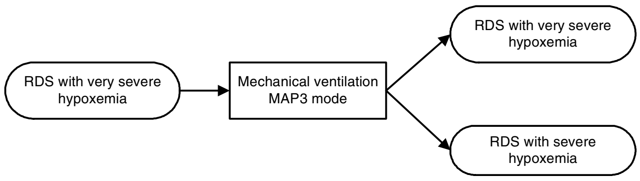

RDS with very severe hypoxemia
Mechanical ventilation MAP3 mode
RDS with very severe hypoxemia
RDS with severe hypoxemia

Fig. 39. The medical planning rule

Let us notice that there exists a certain similarity between the planning rules presented in the subsection and planning operators known from literature (see, *e.g.*, [70]). Similarly to the planning rule each operator describes an action which may be performed on a given complex object. However, each planning operator may have many initial conditions of its execution and many effects of its execution expressed with the help of a family of logical conditions which are to be satisfied after creating the operator. Whereas, in the approach described in this paper all initial conditions of executing a given planning rule are represented using one state which is a complex spatio-temporal concept which requires approximation. Similarly, the effects of planning rule performance are also represented using states which require approximation, which also distinguishes the presented approach from the methods known from literature.

A more complex tool, used to model changes of the states of complex objects as a result of applying action, is a planning graph whose paths describe such changes.

Definition 55 (*A planning graph*).

1. *A planning graph for objects of a fixed type is an ordered triple* $\mathbf{PG} = (S, A, E)$*, where* S *is a nonempty and finite set of states,* A *is a nonempty and finite set of actions and* $E \subseteq (S \times A) \cup (A \times S)$ *is a set of directed edges.*
2. *If* $\mathbf{PG} = (S, A, E)$ *is a planning graph, then any* k*-element sequence* $(v_1, ..., v_k)$ *of elements from the set* $S \cup A$ *such that* $k > 1$ *and* $(v_i, v_{i+1}) \in E$ *for* $i \in \{1, ..., k-1\}$*, is called a path in the planning graph* $\mathbf{PG}$*.*
3. *A family of all paths with length* k *in the planning graph* $\mathbf{PG}$ *is denoted by* $PATH(\mathbf{PG}, k)$*, while a family of all paths in the planning graph* $\mathbf{PG}$ *is denoted by* $PATH(\mathbf{PG})$*.*
4. *Any path* $p' = (v_i, ..., v_j) \in PATH(\mathbf{PG}, j-i+1)$ *created by removing from the path* $p = (v_1, ..., v_k) \in PATH(\mathbf{PG}, k)$ *elements* $v_1, ..., v_{i-1}$ *and* $v_{j+1}, ..., v_k$*, where* $i, j \in \{1, ..., k\}$ *and* $i < j$*, is called a sub-path of the path* p *and is denoted by* $Subpath(p, i, j)$*.*

Let us notice, that from the point of view of automata theory the planning graph is an nondeterministic finite automata in which the automata's states are states from the planning graph, the automata's alphabet is the set of actions from the planning graph and the transfer function is described by the edges of the planning graph.

Such paths in the planning graph are of a particular meaning for the process of automated planning. They tell us how it is possible to bring complex objects from the given state to another given state using actions. Therefore, these types of paths is called *plans.*

Definition 56 (*A plan in a planning graph*). *Let* $\mathbf{PG} = (S, A, E)$ *be a planning graph.*

1. *Any path* $(v_1, ..., v_k) \in PATH(\mathbf{PG}, k)$ *is called a plan in the planning graph* $\mathbf{PG}$ *if and only if* $k > 2$ *and* $v_1, v_k \in S$*.*

2. *A family of all plans with length k in the planning graph* **PG** *is denoted by $PLAN(\mathbf{PG}, k)$, while a family of all plans in the planning graph* **PG** *is denoted by $PLAN(\mathbf{PG})$.*
3. *If $p = (v_1, ..., v_k) \in PLAN(\mathbf{PG}, k)$, then any sub-path $Subpath(p, i, j)$ such that $v_i, v_j \in S$, is called a sub-plan of the plan p and is denoted by $Subplan(p, i, j)$.*

Below, we present an example which illustrates such concepts as: a planning graph, a path in the planning graph and a plan in the planning graph.

Example 35. Let us consider planning graph $\mathbf{PG} = (S, A, E)$ such that $S = \{s_1, s_2, s_3, s_4\}$, $A = \{a_1, a_2, a_3\}$ and $E = \{(s_1, a_1), (s_1, a_2), (s_2, a_3), (s_3, a_3), (a_1, s_1), (a_1, s_3), (a_1, s_4), (a_2, s_1), (a_2, s_2), (a_3, s_4), (a_3, s_2), (a_3, s_3)\}$. This graph is presented in Fig. 40 where the states are represented using ovals, and actions are represented using rectangles. Each link between the nodes of this graph represents a time dependencies. For example, the link between state s_1 and action a_1 tells us that in state s_1 of the complex object action a_1 may be performed, whereas the link between action a_1 and state s_3 means that after performing action a_1 the state of the complex object may change to s_1. An example of a path in graph **PG** is sequence (a_2, s_2, a_3, s_4) whereas path $(s_1, a_2, s_2, a_3, s_3)$ is an exemplary plan in graph **PG**. □

Having the concept of the planning graph defined, the so-called *planning problem* may be defined which works in the way that for a given initial state it should be proposed such a sequence of nodes from the planning graph that brings the initial state to the expected target state. Formally, the planning problem in the elementary version may be depicted in this way.

Problem. *The planning problem* **Input:**

– *planning graph* $\mathbf{PG} = (S, A, E)$,
– *initial state* s_i,
– *target state* s_t.

Output: *Plan* $p = (v_1, ..., v_k) \in PLAN(\mathbf{PG})$ *such that* $v_1 = s_i$ *and* $v_k = s_t$.

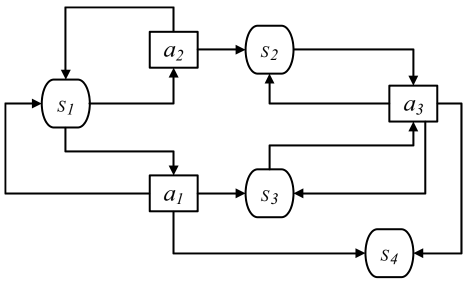


Fig. 40. An exemplary planning graph

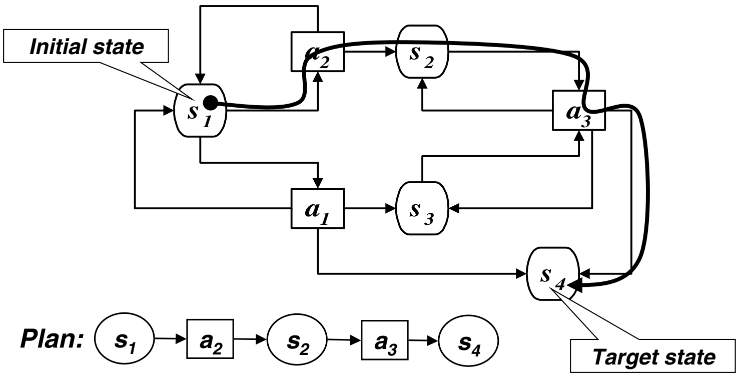


Fig. 41. The output for the planning problem

Fig. 41 presents a solution to the problem of finding a plan bringing state s_1 to state s_4 in the planning graph from Example 35.

The planning graph may be obtained through linking available planning rules, and in order to obtain a planning graph through linking planning rules belonging to a given family of planning rule F, the four following steps should be performed:

1. create a set of states S as a sum of all states which occur in the predecessors and successors of rules of family F,
2. create a set of actions A as a sum of all actions which occur in the rules of family F,
3. create a set of edges E as a sum of all pairs (s, a) for which there exists such a rule in family F that s is the predecessor of this rule and a is an action occurring in this rule,
4. add all the pairs (a, s) to set E for which there exists such a rule in family F that a is the action occurring in this rule and s occurs in the successor of this rule.

In Fig. 42 we present how the three following rules:

- $(s_1, a_1) \rightarrow s_1|s_2$,
- $(s_1, a_2) \rightarrow s_1|s_2$,
- $(s_2, a_1) \rightarrow s_1|s_2$,

may be linked to make a planning graph.

Let's notice that it exists an essential difference between the behavioral graph (see Definition 35 and Definition 53), and the planning graph (see Definition 55). There is one kind of nodes in the behavioral graph only, representing properties of behavior of complex objects during certain period (*e.g.*, time window). Whereas, there are the following two kinds of nodes in the planning graph, namely, states of complex objects (registered in a time point) and actions, that can be performed on complex objects during some period (*e.g.*, time window). Hence, the main

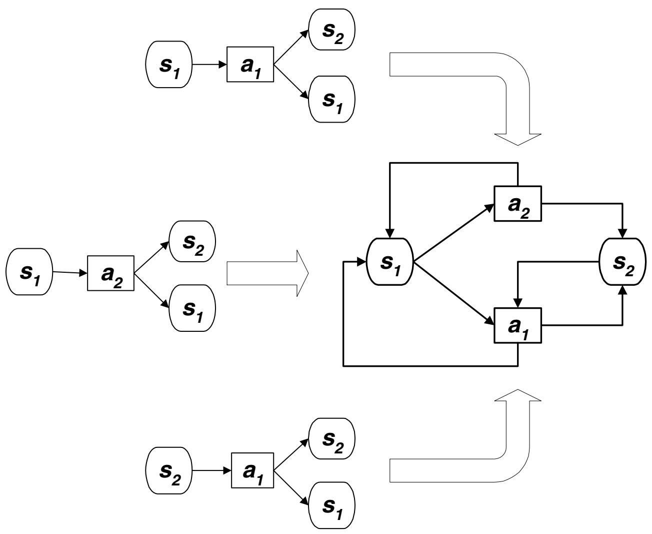


Fig. 42. From planning rules to a planning graph

application of behavioral graphs is to represent observed properties of complex objects, while the main application of planning graphs is to represent changes of object's parameters in the expected direction.

Similarly, there exists a certain similarity here between planning graphs known from literature and planning graphs defined in this paper. It works in the way that in both approaches there occur states, actions and links between them. However, the planning graph known from literature (see, *e.g.*, [70]) is constructed in order to plan the sequence of actions for the established initial state and its construction is aimed at this particular state. Moreover, the construction of this graph is layered and individual layers are connected with the next steps of the plan under construction. However, in this paper the planning graphs are constructed in order to depict the whole knowledge (all possible actions together with their results) concerning the behavior planning of complex objects. Apart from that, there is a difference in understanding states of complex objects (nodes of planning graphs). In approaches known from literature the state of a complex object may be read directly from the values of its parameters or from a simple analysis of dependencies between these values. However, in the approach presented in this paper the state of the complex object is described in a natural language with the help of complex concepts which require approximation (see the beginning of Subsection 7.3).

There also exists a similarity between the concept of the planning graph (see Definition 55) and C/E–systems well known from literature (see, *e.g.*, [350]). The similarity is that both graphs look very similar: the states from the planning graph correspond to the conditions from C/E–system and actions from the behavioral graphs correspond to the events from the C/E–system. However, it should be emphasized here that the interpretation of both graphs is significantly different. In the case of C/E–systems the dynamics of the real system modeled by the net is based on the simulating an occurrence of an event but a given event might have taken place only when all the conditions from which the arches are led to a given event are satisfied. Apart from that, after the occurrence of a given event all conditions, to which the arches of a given event are transferred, are satisfied. It happens differently in the case of the planning graph. The action may be performed when the complex object is in one of the initial states of a given action, that is, in one of the states from which the arches are transferred to this action (complex object may not be simultaneously in more than one states). Similarly, after the performance of the action the complex object goes to one exit state of a given action, which also differentiates significantly the planning graph from C/E–systems. Besides, there is another important difference. In the case of C/E–systems, conditions have a local character, whilst in the case of the planning graph, states have a global character.

7.5 Identification of the Current State of Complex Objects

At the beginning of planning the behavior of the complex object based on a given planning graph, the initial state of this object should be determined. In other words, one of the states occurring in the planning graph should be located in which there is the complex object under examination. In this paper, each state of the planning graph is treated as a spatio-temporal complex concept and to recognize such a state we propose the two following approaches:

1. ask an expert from a given domain to indicate the appropriate state in which there is a given complex object,
2. treat the state as a temporal concept and use methods of temporal concept approximation described in Section 6.

The first of the above possibilities is very convenient, because it does not require the construction of any algorithms. It has, however, a very important drawback: the engagement of an expert in the functioning of the planning system may be too absorbing. However, in some cases the application of this method is possible and sensible. For example, in hospital conditions the current condition of a patient may be determined by an experienced doctor or negotiated by a group of experienced doctors at the established times of the day (*e.g.*, in the morning and in the evening), whereas through the remaining time of the day the treatment of the patient conducted by the doctor on duty may be supported by the planning system and its performance assumes the initial condition of the patient determined by a group of doctors and suggests further treatment.

The second possibility of recognition of the current state of the complex object is treating the state as a temporal concept and using methods of its approximation described in Section 6; and at the same time the interpretation of such a concept is slightly different from the one which appeared in Section 6. In Section 6, the temporal concept described the behavior of a complex object over a certain period of time (time window). Here, however, the temporal concept represents the consequences of the complex object's behavior over a certain period of time, that is, the current state of a complex object (at a time point). We assume that to determine the state of a complex object at a time point, the observation of this object is necessary over a certain period of time (time window). For example, to determine the fact that the patient's condition is very serious, it is often not sufficient to determine what his current medical parameters are (*e.g.*, the results of a clinical interview or his/her laboratory results), but it is necessary to observe how the medical parameters of the patient have changed recently and how the patient's organism reacts to specific treatment methods. It may happen that the patient's medical parameters are very bad, but the application of the typical treatment method causes a sudden and permanent improvement. A crucial modification of methods of temporal concept approximation in the case of state approximation is the usage of information about actions performed on the complex object. This information may be easily introduced to these methods in the form of additional conditional attributes of the c-temporal information system.

Therefore, using methods from Section 6 a stratifying classifier may be constructed for each state, which for a given complex object, provides the degree to which this object belongs to a given state. Next, all these classifiers are linked in order to obtain a general aggregating classifier which recognizes the state of the complex object. Such an aggregating classifier is called *a state identifying classifier.* The performance of the state identifying classifier consists in its choosing such a state for the tested complex object that the stratifying classifier corresponding to this state provided the highest degree of membership for the tested complex object.

7.6 Language of Features of Paths of Planning Graphs

In the further part of the section, we construct information systems whose objects are the paths in the planning graphs and the attributes are the properties (features) of these paths. Therefore, currently we define the *FPPG*-language in which we express features of paths of planning graphs.

Definition 57 (*A language for defining features of paths in planning graphs*). *Let* $\mathbf{PG} = (S, A, E)$ *be a planning graph and let* $\mathbb{N}$ *be a set of natural numbers. A language for defining features of paths in planning graphs (denoted by* $FPPG(\mathbf{PG})$ *or* $FPPG$*-language, when* $\mathbf{PG}$ *is fixed) is defined for the planning graph* $\mathbf{PG}$ *in the following way:*

- *the set* $AL_{FPPG}(\mathbf{PG}) = (2^S \setminus \emptyset) \cup (2^A \setminus \emptyset) \cup \mathbb{N} \cup (0,1] \cup \{$ *Exists, Each, Occurence, First, Last, Order* $\} \cup \{\neg, \vee, \wedge\}$ *is an alphabet of the language* $FPPG(\mathbf{PG})$*,*
- *atomic formulas of the language* $FPPG(\mathbf{PG})$ *are constructed in the following way:*
  1. *for any pair of non-empty sets* $X, Y \subseteq S$*,* $l, r \in \mathbb{N}$ *and* $t \in (0,1]$*, expressions of the form* $First(X,l,r)$*,* $Last(X,l,r)$*,* $Exists(X,l,r)$*,* $Each(X,l,r)$*,* $Occurence(X,l,r,t)$*,* $Order(X,Y,l,r)$ *are atomic formulas of the language* $FPPG(\mathbf{PG})$*,*

2. *for any pair of non-empty sets* $B, C \subseteq A$, $l, r \in \mathbb{N}$ *and* $t \in (0, 1]$, *expressions of the form* $First(B, l, r)$, $Last(B, l, r)$, $Exists(B, l, r)$, $Each(B, l, r)$, $Occurence(B, l, r, t)$, $Order(B, C, l, r)$ *are atomic formulas of the language* $FPPG(\mathbf{PG})$,
3. *for any pair of non-empty sets* $X \subseteq S$ *and* $B \subseteq A$, $l, r \in \mathbb{N}$ *and* $t \in (0, 1]$, *expressions of the form* $Order(X, B, l, r)$ *and* $Order(B, X, l, r)$ *are atomic formulas of the language* $FPPG(\mathbf{PG})$.

Currently, we determine the semantics of the language $FPPG(\mathbf{PG})$. Each formula of the language $FPPG(\mathbf{PG})$ is treated as the description of a set of paths belonging to the set $PATH(\mathbf{PG})$.

Definition 58. *Let* $\mathbf{PG} = (S, A, E)$ *be a planning graph. The semantics of the language FPPG is defined in the following way:*

1. *for any non-empty set* $X \subseteq S$, *numbers* $l, r \in \{1, ..., k\}$ *(where* $l < r$*) and* $t \in (0, 1)$*:*
   - $|Exists(X, l, r)|_{FPPG(\mathbf{PG})} =$
   $$\{(v_1, ..., v_k) \in PATH(\mathbf{PG}) : \exists_{i \in \{l,...,r\}}\ v_i \in X\},$$
   - $|Each(X, l, r)|_{FPPG(\mathbf{PG})} =$
   $$\{(v_1, ..., v_k) \in PATH(\mathbf{PG}) : \forall_{i \in \{l,...,r\}}\ \textit{if}\ v_i \in S\ \textit{then}\ v_i \in X\},$$
   - $|First(X, l, r)|_{FPPG(\mathbf{PG})} =$
   $$\{(v_1, ..., v_k) \in PATH(\mathbf{PG}) : v_l \in X\},$$
   - $|Last(X, l, r)|_{FPPG(\mathbf{PG})} =$
   $$\{(v_1, ..., v_k) \in PATH(\mathbf{PG}) : v_r \in X\},$$
   - $|Occurence(X, l, r, t)|_{FPPG(\mathbf{PG})} =$
   $$\{(v_1, ..., v_k) \in PATH(\mathbf{PG}) : \frac{card(\{i \in \{l, ..., r\} : v_i \in X\})}{card(\{i \in \{l, ..., r\} : v_i \in S\})} \geq t\},$$
2. *for any non-empty set* $B \subseteq A$, $l, r \in \{1, ..., k\}$ *(where* $l < r$*) and* $t \in (0, 1)$*:*
   - $|Exists(X, l, r)|_{FPPG(\mathbf{PG})} =$
   $$\{(v_1, ..., v_k) \in PATH(\mathbf{PG}) : \exists_{i \in \{l,...,r\}}\ v_i \in B\},$$
   - $|Each(X, l, r)|_{FPPG(\mathbf{PG})} =$
   $$\{(v_1, ..., v_k) \in PATH(\mathbf{PG}) : \forall_{i \in \{l,...,r\}}\ \textit{if}\ v_i \in A\ \textit{then}\ v_i \in B\},$$
   - $|First(X, l, r)|_{FPPG(\mathbf{PG})} =$
   $$\{(v_1, ..., v_k) \in PATH(\mathbf{PG}) : v_l \in B\},$$

– $|Last(X,l,r)|_{FPPG(\mathbf{PG})} =$

$$\{(v_1,...,v_k) \in PATH(\mathbf{PG}) : v_r \in B\},$$

– $|Occurence(X,l,r,t)|_{FPPG(\mathbf{PG})} =$

$$\{(v_1,...,v_k) \in PATH(\mathbf{PG}) : \frac{card(\{i \in \{l,...,r\} : v_i \in B\})}{card(\{i \in \{l,...,r\} : v_i \in A\})} \geq t\},$$

3. *for any sets X, Y, B, C (where $X, Y \subseteq S$ and $B, C \subseteq A$) and $l, r \in \{1,...,k\}$ (where $l < r$):*
   – $|Order(X,Y)|_{FPPG(\mathbf{PG})} =$

$$\{(v_1,...,v_k) \in PATH(\mathbf{PG}) : \exists_{i,j \in \{l,...,r\},\ i<j}\ v_i \in X\ \wedge\ v_j \in Y\},$$

   – $|Order(B,C)|_{FPPG(\mathbf{PG})} =$

$$\{(v_1,...,v_k) \in PATH(\mathbf{PG}) : \exists_{i,j \in \{l,...,r\},\ i<j}\ v_i \in B\ \wedge\ v_j \in C\},$$

   – $|Order(X,B)|_{FPPG(\mathbf{PG})} =$

$$\{(v_1,...,v_k) \in PATH(\mathbf{PG}) : \exists_{i,j \in \{l,...,r\},\ i<j}\ v_i \in X\ \wedge\ v_j \in B\},$$

   – $|Order(B,X)|_{FPPG(\mathbf{PG})} =$

$$\{(v_1,...,v_k) \in PATH(\mathbf{PG}) : \exists_{i,j \in \{l,...,r\},\ i<j}\ v_i \in B\ \wedge\ v_j \in X\}.$$

Below, we provide several examples of formulas of the language *FPPG* constructed for the planning graph from Example 35.

– Formula $First(\{s_2\},1,4)$ describes the path whose first state from the node number 1 to node number 4 is state s_2. This is for example path (s_2, a_3, s_3, a_3, s_3).
– Formula $Exists(\{s_2\},2,5)$ describes the path in which, from node number 2 to node number 5 there exists state s_2. This is for example path $(s_1,a_2,s_2,a_3,s_3,a_3)$.
– Formula $Exists(\{s_2,s_3\},2,7)$ describes the path in which, from node number 2 to node number 5 there exists state s_2 or state s_3 or both of them. There are for example paths: $(s_1,a_2,s_2,a_3,s_3,a_3,s_2,a_3)$, $(s_1,a_1,s_3,a_3,s_3,a_3,s_3,a_3,s_3)$ or $(s_1,a_2,s_1,a_2,s_2,a_3,s_2,a_3,s_2)$.
– Formula $Each(\{a_3\},1,5)$ describes the path, in which from node number 1 to node number 5 there is only action a_3. This is for example path (s_2,a_3,s_3,a_3,s_2).
– Formula $Occurence(\{s_2\},3,7,0.6)$ describes the path, in which from node number 3 to node number 7, at least 60% of all states constitute state s_2. This is for example path number $(s_1,a_2,s_2,a_3,s_3,a_3,s_2,a_3,s_4)$.
– Formula $Order(\{a_2\},\{a_3\},2,7)$ describes the path, in which from node number 2 to node number 7, firstly action a_2 is performed and then action a_3. This is for example path $(s_1,a_2,s_2,a_4,s_3,a_3,s_3)$.

– Formula $Order(\{a_2\}, \{s_3\}, 2, 7)$ describes the path, in which from node number 2 to node number 6, firstly action a_2 is performed and then state s_3 is observed. This is for example path $(s_1, a_2, s_1, a_4, s_2, a_3, s_3)$.

Patterns of the language $FPPG$ may be applied in defining the path properties in the planning graph. Owing to this each path in the planning graph may be represented using the values of its features. It enables approximation of the concepts determined in the set of paths with the help of classifiers (see Section 7.7).

7.7 Resolving Table

As we mentioned before, the output for the planing problem for a single complex object is a path in the planning graph from the initial node-state to the expected (target) node-state (see Fig. 41).

However, in the planning graphs there often appears a problem of non-deterministic choice of one of the actions possible to apply in a given state. For example, in the graph from Fig. 40 action a_1 or a_2 may be performed in state s_1. Apart from that, there also occurs the uncertainty concerning the choice of the state after applying the action. For example, in the graph from Fig. 40 in state s_1 after applying action a_2 the complex object may change to state s_2 or remain in state s_1. That is why there may be usually many solutions to a given planning problem consisting in going from the initial state to the target state on different paths in the graph. Assuming that we always treat all actions and states in the same way and the choice of actions in a given state and the choice of the state after applying the action is random or directed using a heuristic function onto the target state, then to solve the planning problem one may use planning methods known from literature such as: *forward search*, *backward search* or *heuristic for state-space search*, which in fact would consist in searching the planning graph (see, *e.g.*, [70, 76]).

However, in practice there often occurs such a situation that the automatically generated plan must be compatible with the plan suggested by an expert (*e.g.*, the treatment plan should be compatible with the plan suggested by human experts from a medical clinic). Therefore, it is strongly recommended that the method of the verification and evaluation of generated plans should be based on the similarity between the generated plan and the plan proposed by human experts (see Section 7.21). Apart from that we need tools which during the generating the automatic plan may be used to solve the conflicts occurring between actions which may be used at a given planning stage in such a way as to make this choice compatible with domain knowledge provided by the experts. Such tools should work on the basis of the current state of the tested complex object and on the basis of information about the previously observed states of the tested complex object as well as on the basis of information about actions performed earlier on this object. In other words, while choosing the action needed to perform in a given state of the complex object one has to use information about the sequence of states and actions which have led the object under examination to the current state. In terms of the planning graph such information is simply a path of an established length k in this graph, which ends in the current state of the complex object under examination. In practice, for a given state from the planning graph there exist very many different paths which end exactly with that state. That is why in constructing tools allowing choosing actions on the basis of the path in the planning graph preceding the current state of the complex object, one should use the available data sets gathered during the observation of the complex dynamical systems. So far, we have used temporal information systems to represent such data sets. However, in the temporal information system the actions performed on the complex objects are not represented in an overt way although they may obviously be represented using the established attribute. Therefore, we define a certain particular type of a temporal information system which is called *a temporal information system with actions.*

Definition 59 (*A temporal information system with actions*)**.** *A temporal information system with actions is a seven-element tuple:*

$$\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t}, a_c),$$

where a tuple $(U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t})$ *is the temporal information system and* $a_c \in A$ *is a distinguished attribute in the set* A *different from attributes* a_{id} *and* a_t*, which is an attribute identifying the action performed at a time point represented by a given object of system* $\mathbf{T}$*.*

Thus, in the temporal information system with actions for each object of this system $u \in U$ (at a time point of this system) the action performed at this point is remembered and it is action $a_c(u)$.

Let us notice that we consider here one action performed on the complex object at a given time point. However, it seems that in practical applications, at a given time point a sequence of actions could be performed synchronically on the complex object. However, they are always actions chosen from the established set of single actions. Therefore, the action performed at a given time point may be treated as a subset of the established set of single actions. For example, if $M = \{m_1, m_2, m_3, m_4\}$ is a set of medicaments which may be used during the treatment of a certain illness, then an example of a specific action of the patient's treatment is action $\{m_1, m_3\}$ which consists in giving the patient medicine m_1 and m_2 simultaneously. Apparently, other actions are also possible. For example, action $\{m_1, m_2, m_3, m_4\}$ is the action of giving all possible medicaments. While, action $\{\ \}$ (empty set) is the action of not giving any medicament.

For temporal information system with actions, one can speak about states in which there are individual objects of this system. We mean here the states specified in a planning graph. However, the identification of the state of a given time point requires application of the classifier constructed specially for this purpose (see Section 7.5).

If it is possible to identify the current state of the examined object and the actions performed at individual time points are known, then it is possible to represent the history of the examined complex object arranged as a path from the planning graph observed in a given temporal information system.

Definition 60. *Let us assume that:*

- $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t}, a_c)$ *is a temporal information system with actions,*
- $\mathbf{PG} = (S, A, E)$ *is a planning graph,*
- $s \in S$ *is a fixed state,*
- k *is a fixed plan length in the planning graph* $\mathbf{PG}$.

1. *Any plan* $p = (v_1, ..., v_k) \in PLAN(\mathbf{PG}, k)$ *is called a plan occurring in system* $\mathbf{T}$ *if exists such a time window* $W = (u_1, ..., u_l) \in TW(\mathbf{T}, l)$ *such that* $l = \frac{k-1}{2} + 1$*) and at the successive time points of this window there occur states* $v_1, v_3, v_5..., v_k$ *and at the time points from* $u_1, ..., u_{l-1}$ *actions* $v_2, v_4, ..., v_{k-1}$ *are performed respectively. For a given plan p such a time window is called a time window of this plan.*
2. *A set of all time windows of a given plan p in system* $\mathbf{T}$ *is denoted by* $TW(\mathbf{T}, p)$.
3. *A set of all plans, occurring in system* $\mathbf{T}$ *of the length k is denoted by* $DPLAN(\mathbf{T}, \mathbf{PG}, k)$.
4. *A set of all plans, occurring in system* $\mathbf{T}$ *and ending with state s and of the length k, is denoted by* $DPLAN(\mathbf{T}, \mathbf{PG}, s, k)$.

Let us notice that set $DPLAN$ may be determined in such a way that firstly a set of all time windows for a given temporal information system is determined (see Section 6.7) and then these windows are treated as potential time windows of plans from the set $DPLAN$.

Using the planning graph paths occurring in data, decision tables may be constructed and what follows there may also be constructed classifiers which allow solving conflicts between actions which may be performed in a given state for the complex object. Let us notice that the classifiers mentioned above also allow determining what the state of the complex object will be after performing the chosen action. The starting point for the construction of such classifiers is *a resolving table* (see Fig. 43).

Definition 61 (*A resolving table*)**.** *Let us assume that:*

- $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t}, a_c)$ *is a temporal information system with actions,*
- $\mathbf{PG} = (S, A, E)$ *is a planning graph,*
- $s \in S$ *is a fixed state,*
- k *is a fixed length of path,*
- $\Phi = \{\phi_1, ..., \phi_m\} \subseteq FPPG(\mathbf{PG})$ *is a family of formulas defined by experts,*
- $\mathcal{P}_{FPPG} = (U, \Phi, \models_{FPPG})$ *is a property system, where*

  $$U = DPLAN(\mathbf{T}, \mathbf{PG}, s, k)$$

  and the satisfiability relation $\models_{FPPG} \subseteq U \times \Phi$ *is defined in the following way:* $\forall p = (v_1, ..., v_k) \in U$ *and* $\phi \in \Phi$:

  $$u \models_{FPPG} \phi \quad \Leftrightarrow \quad Subpath(p, 1, k-2) \models_{FPPG(\mathbf{PG})} \phi.$$

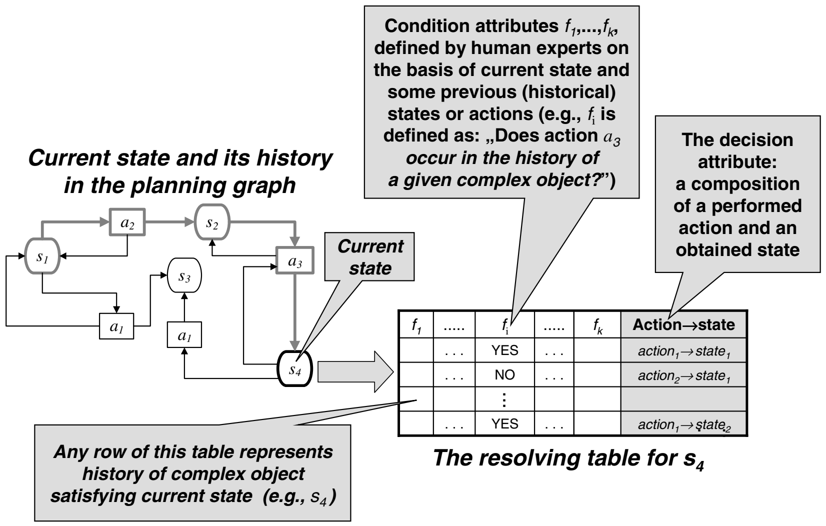

Condition attributes $f_1,\ldots,f_k$, defined by human experts on the basis of current state and some previous (historical) states or actions (e.g., f_i is defined as: „Does action a_3 occur in the history of a given complex object?")
Current state and its history in the planning graph
The decision attribute: a composition of a performed action and an obtained state
Current state
f_1 ..... f_i ..... f_k Action→state
YES action$_1$→state$_1$
NO action$_2$→state$_1$
YES action$_1$→state$_2$
Any row of this table represents history of complex object satisfying current state (e.g., s_4)
The resolving table for s_4

Fig. 43. The scheme of construction of the resolving table for a given state

A resolving table for the state s from the planning graph **PG** *constructed using the length of path k is a decision table* $\mathbf{RT}(s,k) = (U, A, d)$*, where:*

– (U, A) *is an information system defined be the property system* $\mathcal{P}_{FPPG}$*,*
– *d is a decision attribute, where values of the attribute d, being ordered pairs of the form* $(action, state)$*, are computed in the following way:*

$$\forall p = (v_1, ..., v_k) \in U : d_i(p) = (v_{k-1}, v_k).$$

The objects of this table are paths in the planning graph observed in data, starting and ending with a state. Thus, they are plans. Conditional attributes describe the properties of these paths excluding the last two nodes of each path and they are defined on the basis of the formulas of the language *FPPG* provided by the expert, whereas the values of the decision attribute are arrangement of the action performed after the last but one state on the path and the last state on the path.

Example 36. In Fig. 44, the planning graph for the RDS treatment is shown. For each state occurring in the graph, with the use of available data sets concerning the treatment of respiratory failure, resolving tables may be constructed. Conditional attributes of these tables are created with the use of patterns defined by experts in language *FPPG*. Below, we present examples of typical patterns of this type:

1. the first (last) state in the plan is the *RDS excluded* (*RDS with mild hypoxemia*, *RDS with severe hypoxemia*, *RDS with very severe hypoxemia*, *RDS with mild or severe hypoxemia*, *RDS with severe or very severe hypoxemia*) state,

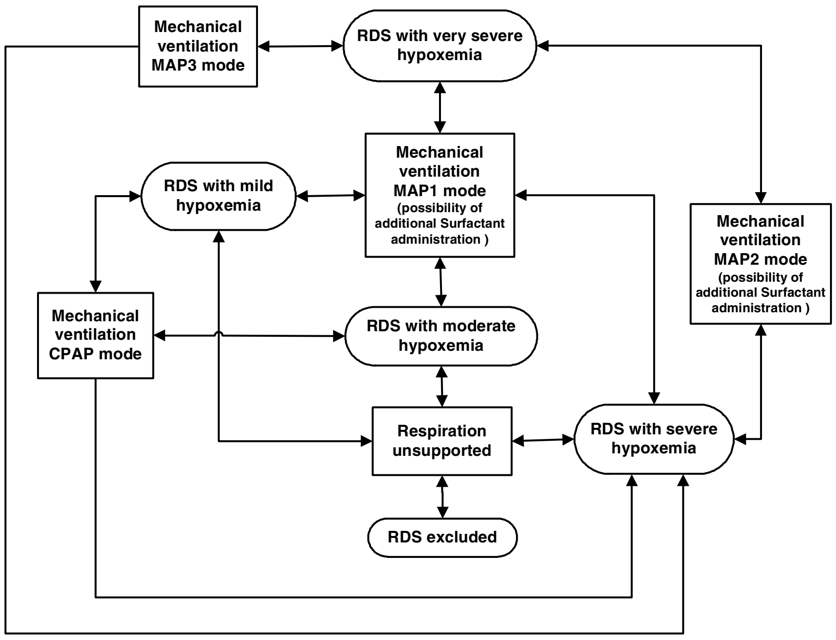

Mechanical ventilation MAP3 mode
RDS with very severe hypoxemia
RDS with mild hypoxemia
Mechanical ventilation MAP1 mode (possibility of additional Surfactant administration)
Mechanical ventilation MAP2 mode (possibility of additional Surfactant administration)
Mechanical ventilation CPAP mode
RDS with moderate hypoxemia
Respiration unsupported
RDS with severe hypoxemia
RDS excluded

Fig. 44. A planning graph for the treatment of infants during the RDS

2. the first (last) action in the plan is the *Mechanical ventilation MAP1 mode*[3] (*Mechanical ventilation MAP2 mode*, *Mechanical ventilation MAP3 mode*, *Mechanical ventilation CPAP mode*[4], *Respiration unsupported*, *Mechanical ventilation MAP2 or MAP3 mode*) action,

[3] Invasive mechanical ventilation is a method to mechanically assist or replace spontaneous breathing when patients cannot do so on their own. It is administered after an invasive intubation, a procedure wherein an endotracheal or tracheostomy tube is inserted into the airway, through which air is directly delivered under pressure (see [328] for more details). It could be simplify, that *mean airway pressure* (MAP) delivered by mechanical device is proportional to severity of respiratory failure. For purpose of our experiments mechanical ventilation was divided into three following modes:
– *MAP1* - airway pressure lower than 10 cm H2O (low-intensity ventilation),
– *MAP2* - airway pressure 10-16 cm H2O (middling-intensity ventilation),
– *MAP3* - airway pressure higher than 16 cm H2O (high-intensity ventilation).

[4] *CPAP* (continuous positive airway pressure) - a method of non-invasive ventilation delivering a stream of compressed air via a hose to a nasal pillow, nose mask or full-face mask, splinting the airway (keeping it open under air pressure). This is a gentle type of respiratory ventilation, which can prevent the need for endotracheal intubation, or allow earlier extubation of critically ill patients (see [328] for more details).

3. in the plan there occurs the *RDS excluded* (*RDS with mild hypoxemia*, *RDS with severe hypoxemia*, *RDS with very severe hypoxemia*, *RDS with mild or severe hypoxemia*, *RDS with severe or very severe hypoxemia*) state,
4. in the plan there occurs the *Mechanical ventilation MAP1 mode* (*Mechanical ventilation MAP2 mode*, *Mechanical ventilation MAP3 mode*, *Mechanical ventilation CPAP mode*, *Respiration unsupported*, *Mechanical ventilation MAP1 or CPAP mode*) action,
5. in the plan there occurs only the *RDS excluded* (*RDS with mild hypoxemia*, *RDS with severe hypoxemia*, *RDS with very severe hypoxemia*, *RDS with mild or severe hypoxemia*, *RDS with severe or very severe hypoxemia*) state,
6. in the plan there occurs only the *Mechanical ventilation MAP1 mode* (*Mechanical ventilation MAP2 mode*, *Mechanical ventilation MAP3 mode*, *Mechanical ventilation CPAP mode*, *Respiration unsupported*) action,
7. the *RDS with very severe hypoxemia* state occurs in the 70% of states of the plan,
8. from the middle of the plan to its end in 80% of the states there occurs the *RDS excluded* state,
9. if there occurs the *RDS with mild hypoxemia* state in the plan then the *RDS with severe hypoxemia* state occurs in the further part of this plan,
10. if there occurs the *Mechanical ventilation MAP3 mode* action in the plan then the *Mechanical ventilation MAP2 mode* action occurs in the further part of this plan,
11. if there occurs the *Mechanical ventilation MAP2 mode* action in the plan then *RDS with mild hypoxemia* state occurs in the further part of this plan.

□

A classifier may be constructed for the resolving table which we call *a resolving classifier.*

Definition 62 (*A resolving classifier*)**.** *Let us assume that:*

- $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t}, a_c)$ *is a temporal information system with actions,*
- $\mathbf{PG} = (S, A, E)$ *is a planning graph,*
- $s \in S$ *is a fixed state,*
- k *is a fixed length of path,*
- $\mathbf{RT}(s, k) = (U, A, d)$ *is a resolving table for the fixed state s from the planning graph* $\mathbf{PG}$ *constructed using the length of path k.*

1. *Each stratifying classifier constructed for table $\mathbf{RT}(s, k)$ is called a resolving classifier and is denoted in general by $\mu_{\mathbf{RT}(s,k)}$. This classifier serves the classification of paths which belong to the $DPLAN(\mathbf{T}, \mathbf{PG}, s, k-2)$.*
2. *For any $p \in DPLAN(\mathbf{T}, \mathbf{PG}, s, k-2)$ and resolving classifier $\mu_{\mathbf{RT}(s,k)}$ by $PairList(\mu_{\mathbf{RC}(s,k)}(p))$ we denote a list of pairs $(action, state)$ which the classifier $\mu_{\mathbf{RC}(s,k)}$ returns to the path p ordered in a decreasing order in relation to weigh values proposed by the classifier for all pairs, and at the same time only pairs with non-zero weight are returned.*

3. *If L is a $PairList(\mu_{\mathbf{RC}(s,k)}(p))$, then the i-th pair of this list is marked as $L[i]$. The first element of the pair $L[i]$ (action) is marked as $L[i].action$, whereas the second element of this pair (state) is marked as $L[i].state$.*

Such resolving classifiers can be constructed for all states, i.e., for all associated resolving tables. In addition, these classifiers make it possible to obtain a list of actions and states after usage of actions with their weights in descending order. This is possible using the stratifying classifier.

7.8 Algorithms of Automated Planning for Complex Objects

In the present subsection, we provide three algorithms of automated planning of the complex object behavior. The first one determines one plan of established length starting with the established initial state, and at the same time the final (target) state is not established.

The second algorithm determines the plan starting with the established initial state and ending with the established final state. The length of the generated plan is limited from the top by the established constant value.

The third algorithm, however, determines the list of plans starting with the established (for all plans) initial states, and at the same time the length of the generated plans is established. Similarly to the second algorithm, the length of the generated plan is limited from the top by the established constant value.

The first of the algorithms mentioned above is similar to the algorithm *Forward-search* known from literature (see, *e.g.*, [70, 76]), but instead of choosing randomly the actions to perform in a given state the algorithm goes to the next state on the basis of the decision obtained from the classifier solving conflicts between actions in a given state. Therefore, this algorithm is called *Expert forward search* (see Algorithm 7.1).

However, we assume that during execution of algorithms presented in this section the following elements should by available:

– a planning graph $\mathbf{PG} = (S, A, E)$ for complex objects of a fixed type T,
– a fixed length of path k in the planning graph $\mathbf{PG}$,
– a resolving classifiers $\mu_{\mathbf{RC}(s,k)}$ for all $s \in S$.

The Algorithm 7.1 starts the planning from path $h \in DPLAN(\mathbf{T},\mathbf{PG}, k)$ which describes the previous states of the complex object and the actions applied for this object. Next, using the resolving classifier $\mu_{\mathbf{RC}(s,k)}$, the most appropriate pairs: *state+action* are generated in the next iterations until the plan of the expected length is obtained.

If it is assumed that each of classifiers $\mu_{\mathbf{RC}(s,k)}$ can classify paths for each $s \in S$ within the time of order $O(C)$, where C is a certain constant, then the time complexity of the Algorithm 7.1 is of order $O(n)$, where n is the length of the generated plan.

The Algorithm 7.1 is, on the one hand very fast, but on the other its final result does not always comply with our expectations. For example, in planning for a single complex object we usually wish our planning algorithm to find a

Algorithm 7.1. Expert forward search (EFS)

Input:
- path $h = (h_1, ..., h_k)$ in the planning graph **PG** representing history of a given complex object, that is finished by an initial state to start of automated planning,
- expected length l_p of generated plan.

Output: The plan p generated for the given complex object

```
1  Procedure $EFS(h, l_p)$
2  begin
3      $p :=$ "empty plan"
4      $s := GetLastElementFrom(h)$
5      $p := p + s$ // Add $s$ to the end of the plan $p$
6      while $length(p) < l_p$ do
7          $L := PairList(\mu_{\mathbf{RC}(s,k)}(h))$
8          $p := p + L[1].action + L[1].state$
9          $RemoveFirstTwoElementsFrom(h)$
10         $h := h + L[1].action + L[1].state$
11         $s := L[1].state$;
12     end
13     return $p$
14 end
```

plan which leads the complex object to the established target state. Meanwhile, the final state of the planning using the algorithm EFS depends on classifier $\mu_{\mathbf{RC}(s,k)}$ and cannot be imposed. Therefore, we define the algorithm $EEFS$ which determines the plan starting with the established initial state and ending with the established final state (see Algorithm 7.2).

The Algorithm 7.2 works in such a way that at the stage of planning of a single action, its different variants are taken into consideration which may be performed in a given state. However, for regulation of computational time duration limitation, the value *ActionLimit* is used, that is, limitation of the number of actions which may be performed in a given state (see line 7). Thus, the classifier $\mu_{\mathbf{RC}(s,k)}$ returns the list of pairs (*action* + *state*) sorted decreasingly in relation to the weights obtained from classification, the actions most recommended by classifier $\mu_{\mathbf{RC}(s,k)}$ are always taken into consideration. In this way the algorithm constructs a certain type of a plan tree whose root is the initial state and the leaves are the states after performing the individual variants of the plan. If, during construction of this tree the final state appears, then the work of the algorithm is ended and as a solution a sequence of states and actions is returned which starts in the tree root and ends with the final state that is found. If, during the construction of the plan tree the algorithm does not encounter the

Algorithm 7.2. Exhaustive expert forward search (*EEFS*)

Input:
- path $h = (h_1, ..., h_k)$ in the planning graph **PG** representing history of a given complex object, that is finished by an initial state to start of automated planning,
- target state of planning s_t,
- maximal length of generated plan l_p.

Output: The plan p generated for the investigated complex object, ended by the state s_t

```
Procedure $EEFS(h, s_t, l_p)$
begin
    $p :=$ "empty plan"
    $s := GetLastElementFrom(h)$
    $p := p + s$ // Add $s$ to the end of the plan $p$
    $L := PairList(\mu_{\mathbf{RC}(s,k)}(h))$
    for $i = 1$ to $ActionLimit$ do
        $p_1 := Copy(p)$
        $p_1 := p_1 + L[i].action + L[i].state$
        if $(L[i].state = s_t)$ then return $p_1$
        if $(l_p > 1)$ then
            $h_1 := Copy(h)$
            $RemoveFirstTwoElementsFrom(h_1)$
            $h_1 := h_1 + L[i].action + L[1].state$
            $p_2 := EEFS(h_1, s_t, l_p - 1)$
            if ($p_2$ is not empty ) then
                return $p + p_2$
            end
        end
    end
    return "empty plan"
end
```

final state, then an empty plan is returned which means that the algorithm has not found a solution.

The procedure *EEFS* from the Algorithm 7.2 is recurrent. It is easy to notice that its time complexity is determined by a recurrent equation:

$$T(n) = \begin{cases} A \cdot m + B & \text{for } n = 1 \\ m \cdot T(n-1) + C \cdot m + B & \text{for } n > 1, \end{cases}$$

where A, B and C are certain constants, n is the duration of the plan under construction and m is limitation *ActionLimit*, that is, limitation of the number

Algorithm 7.3. Full exhaustive expert forward search ($FEEFS$)

Input:
- path $h = (h_1, ..., h_k)$ in the planning graph **PG** representing history of a given complex object, that is finished by an initial state to start of automated planning,
- length of generated plan l_p.

Output: The list of plans for the investigated complex object

```
Procedure $FEEFS(h, l_p)$
begin
    $plist$ := "empty list of plans"
    $p$ := "empty plan"
    $s := GetLastElementFrom(h)$
    $p := p + s$ // Add $s$ to the end of the plan $p$
    $L := PairList(\mu_{\mathbf{RC}(s,k)}(h))$
    for $i = 1$ to $ActionLimit$ do
        $p_1 := Copy(p)$
        $p_1 := p_1 + L[i].action + L[i].state$
        if $(l_p > 1)$ then
            $h_1 := Copy(h)$
            $RemoveFirstTwoElementsFrom(h_1)$
            $h_1 := h_1 + L[i].action + L[i].state$
            $plist_1 := FEEFS(h_1, l_p - 1)$
            for $j := 1$ to $Length(plist_1)$ do
                $p_2 := plist_1[j]$
                $p_3 := p_1 + p_2$
                Add plan $p_3$ to the end of the list $plist$
            end
        else
            Add plan $p_1$ to the end of the list $plist$
        end
    end
    return $plist$
end
```

of actions which may be performed in a given state. The solution of the above recurrent equation is the following:

$$T(n) = A \cdot m^n + B \cdot m^{n-1} + C \cdot m \cdot \frac{m^{n-1} - 1}{m - 1} + B \cdot \frac{m^{n-1} - 1}{m - 1}.$$

Thus, the pessimistic time complexity of the procedure $EEFS$ is of order $O(m^n)$. Such a high pessimistic time complexity means that the effective application of this algorithm for non-trivially small n and m is practically impossible.

Therefore, it may be applied only in constructing very short plans with very small values m.

In the task of constructing a plan executing a meta-action for a structured object another planning algorithm for a single object is necessary (see Section 7.15). Namely, in this case the planning target state is also not known, but one has to generate all sensible (compatible with domain knowledge) plans of a given duration for a given complex object. Therefore, we define the algorithm $FEEFS$ (see Algorithm 7.3).

It is easy to notice that the analysis of time complexity of the Algorithm 7.3 is very similar to the case of Algorithm 7.2. Therefore, the pessimistic time complexity of the $FEEFS$ is of order $O(m^n)$ where n is the duration of the plan under construction and m is limitation of the number of actions which may be performed in a given state. This means, that similarly to the case of algorithm $EEFS$ the effective application of algorithm $FEEFS$ for non-trivially small n and m is practically impossible. Therefore, this algorithm is used only for construction of short plans for the need of planning of single meta-actions (see Section 7.15).

7.9 Partial Reconstruction of Plan

Having constructed the plan for a complex object, its execution may take place. For example, let us assume that for a certain complex object the plan (s_1, a_1, ..., a_{i-1}, s_i, a_i, ..., s_n, a_n, s_{n+1}) was constructed which consists of n actions $a_1, ..., a_n$ and $n + 1$ states $s_1, ..., s_{n+1}$. The initial state in this plan is state s_1 and the target state is the state s_{n+1}. The execution of this plans works in such a way that after having identified the current state of the complex object as state s_1, actions from the plan are performed successively, with the changing states of object, until we reach target state s_{n+1}. However, in practice it is not always possible to execute the whole plan. It may, happen that during the execution of the plan such a state of an object appeared that is not compatible with the state proposed by the plan. For example, let us assume that s'_i is such a state which appeared instead of state s_i (see Fig. 45). Then, a question arises whether the execution of the plan should be continued or whether it should be reconstructed (changed). If state s'_i differs slightly from state s_i, then maybe the execution of the current state may be continued. If, however, state s'_i differs significantly from state s_i, then the current plan has to be reconstructed. It would seem that the simplest way to reconstruct a plan is to construct a new one, which starts in state s'_i and ends in target state s_{n+1}. Such a method of reconstruction we call a *total reconstruction*. However, in practical applications a total reconstruction may turn out to be too costly in terms of computation. Therefore, we propose a different method of plan reconstruction which is called *a partial reconstruction*. It consists in constructing a short so-called *repair plan* which brings the complex object to such state s_j that appears in the current state between the states $s_i, ..., s_{n+1}$. On the basis of the repair plan the reconstruction of the current plan is carried out by replacing its fragment beginning at s_i and ending at state s_j with the repair plan.

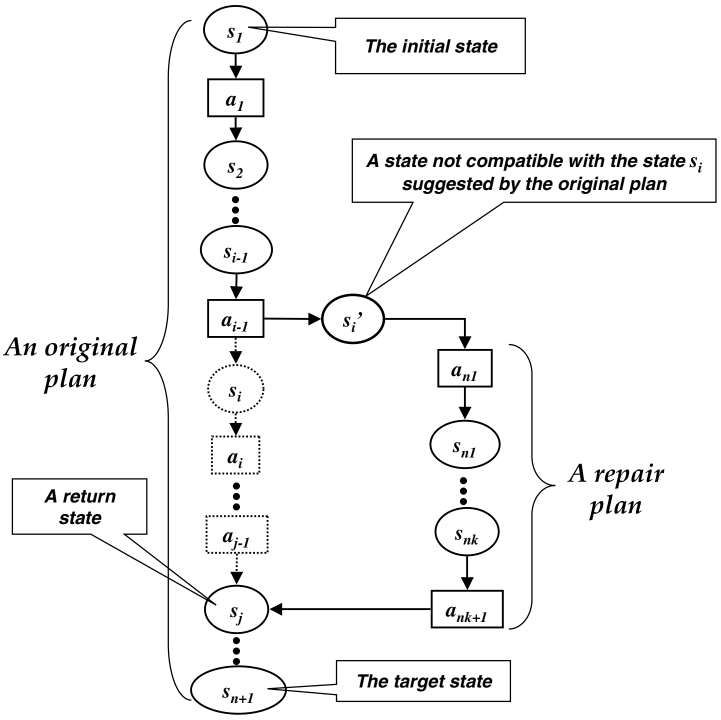


Fig. 45. The partial reconstruction of a plan

Of course the shorter the repair plan is, the more effective a partial reconstruction is in terms of time. If, however, state s_i' differs significantly from state s_i, then finding a short repair plan is impossible and a total reconstruction is the only solution.

We still have to face the problem of estimating the degree to which two states are different. It needs to be done in order to enable the determination when two states differ slightly, differ significantly or differ very much one from another.

There is yet another problem lying in the fact that the difference between two states in the context of plan execution depends not only on those states but also on the context in which those two states are compared, that is, on the fragment of the plan that has been carried out so far, together with the states that have appeared during the current plan execution (these states might have slightly differed from the states described in the plan). Therefore, the estimation of the difference between the plans should be made on the basis of the history of both states (the states and actions performed on the complex object over the period of time preceding the examination of the difference). Formally, the degree to which two states differ can be expressed with the help of the so-called *a function of dissimilarity between of states.*

Definition 63. *Let* $\mathbf{PG} = (S, A, E)$ *is a planning graph for complex objects of a fixed type* T *and* k *is a fixed length of paths in the graph* $\mathbf{PG}$. *Each function:*

$$DISM_{\mathbf{PG}} : PATH(\mathbf{PG}, k) \times PATH(\mathbf{PG}, k) \longrightarrow \{high, moderate, low\}$$

is called a function of dissimilarity between states from the planning graph $\mathbf{PG}$.

The values of the function $DISM_{\mathbf{PG}}$ which belong to set $\{high, moderate, low\}$ are proposed by the expert on the basis of domain knowledge. Value *high* means a high dissimilarity between states, value *moderate* means a moderate dissimilarity between states and value *low means* a low dissimilarity between states.

The definition of a specific function of dissimilarity between states may be given in an overt form, that is, using an expression calculating the value of dissimilarity function. It often happens, however, that experts from a given field are not able to present such an expression and limit themselves to presenting a set of examples of the values of that function, that is, a set of pairs of paths ended with compared states, labelled with the value of the dissimilarity function between states. In this case defining the dissimilarity function requires its approximation using a classifier; and at the same time to define the features of the paths preceding the compared states one may use a family of concepts of a specific ontology constructed for the comparison of the paths. The classifier approximating the function of the dissimilarity between states is called a classifier of dissimilarity between states.

Definition 64. *Let us assume that:*

- $\mathbf{PG} = (S, A, E)$ *is a planning graph for complex objects of a fixed type* T,
- k *is a fixed length of path in the planning graph* $\mathbf{PG}$,
- *a family of concepts* $C_1, ..., C_m \subseteq PATH(\mathbf{PG}, k) \times PATH(\mathbf{PG}, k)$, *which have been defined by experts in order to describe difference aspects of similarity between plans,*
- *a function of dissimilarity between states* $DISM_{\mathbf{PG}}$.

1. *A table of dissimilarity between states from the planning graph* $\mathbf{PG}$ *is a decision table* $\mathbf{DIT}_{\mathbf{PG}} = (U, A, d)$, *where:*
   - $U \subseteq PATH(\mathbf{PG}, k) \times PATH(\mathbf{PG}, k)$,
   - $A = \{a_1, ..., a_m\}$ *is a set of attributes created on the basis of concepts* $C_1, ..., C_m$, *where for any* $i \in \{1, ..., m\}$ *values of* a_i *are computed in the following way:*

$$\forall (p_1, p_2) \in U : a_i\left((p_1, p_2)\right) = \begin{cases} 1 \text{ if } (p_1, p_2) \in C_i \\ 0 \text{ otherwise} \end{cases},$$

   - d *is a decision attribute, where values of the attribute* d *are computed in the following way:*

$$\forall (p_1, p_2) \in U : d(\ (p_1, p_2)\ ) = DISM_{\mathbf{PG}}(\ (p_1, p_2)\ ).$$

2. If $\mathbf{DIT_{PG}} = (U, A, d)$ *is a table of dissimilarity between states from the graph* $\mathbf{PG}$*, then each classifier for the table* $\mathbf{DIT_{PG}}$ *is called a classifier of dissimilarity between states from the graph* $\mathbf{PG}$ *and is denoted in general by* $\mu_{DIT(\mathbf{PG})}$.

Let us notice that not all possible pairs of paths from the set $PATH(\mathbf{PG}, k) \times PATH(\mathbf{PG}, k)$ occur in the table of dissimilarity between states from the planning graph, but only a certain chosen subset of this set. In practice, this limitation is needed because the number of pairs of product $PATH(\mathbf{PG}, k) \times PATH(\mathbf{PG}, k)$ may be so large that the expert is not able to provide all values of decision attribute d for them. That is why in the table of dissimilarity between states there are usually only pairs chosen by an expert, which represent typical cases of determining the function of dissimilarity between states which may be generalized using a classifier.

Now, we may present the algorithm simulating the execution of the plan which foresees the reconstruction of the plan during its execution (see Algorithm 7.4).

However, we assume that during execution of algorithms presented in this section the following elements should by available:

– a planning graph $\mathbf{PG} = (S, A, E)$ for complex objects of a fixed type T,
– a fixed length of path k in the planning graph $\mathbf{PG}$,
– resolving classifiers $\mu_{\mathbf{RC}(s,k)}$ for all $s \in S$,
– a classifier of dissimilarity $\mu_{DIT(\mathbf{PG})}$ between states from the planning graph $\mathbf{PG}$.

The Algorithm 7.4 simulates the execution of the plan found earlier for the complex object. The simulation is performed based on the procedure *Simulate* which on the input takes the history of the current state together with its description of the current state and the action which is to be performed, and on the output the algorithm returns the state which is the effect of this action's application. Although it is possible to imagine this type of procedure as a part of a simulator of the behavior of the complex object (*e.g.*, a traffic simulator, illness development simulator), in this paper by this procedure we mean the changes in the real complex dynamical system which may be triggered by performing particular actions (*e.g.*, changes of the location of the vehicle, changes in the patient's states during treatment et.)

The Algorithm 7.4 uses the reconstruction procedure. Therefore, we present a plan reconstruction algorithm (see Algorithm 7.5).

The Algorithm 7.5 tries to find a short repair plan p_2 (not longer than l_p), which leads the initial state of the reconstruction (the last state in history h) to a state occurring in plan p_1 starting with position pos until reaching position $pos + 2 \cdot (d_r - 1)$. The maximum depth of reconstruction d_r is, thus, the number of states in plan p_1 (starting with the state in position pos), which the algorithm tries to reach with the help of the repair plan. The repair plan is searched with algorithm $EEFS$ (see Section 7.8), although it is possible to apply other planning algorithms that have at least two following parameters: the target state and the maximum duration of the created plan.

Algorithm 7.4. The simulation of the plan with reconstruction

Input:
- path $h = (h_1, ..., h_k)$ in the planning graph **PG** representing history of a given complex object, that is finished by an initial state to start of simulation,
- plan p generated for a given complex object,
- maximal depth d_r of the plan reconstruction,
- maximal length l_p of a repair plan during the reconstruction.

Output: The executed plan p

```
begin
    if ($length(p) < 3$) then
        return "plan p is too short for execution"
    end
    $h_s := Copy(h)$; $h_p := Copy(h)$
    $i := 3$
    while ($i < length(p)$) do
        $s := Simulate(h_s, p[i-1])$
        $RemoveFirstTwoElementsFrom(h_s)$
        $h_s := h_s + p[i-1] + s$
        $RemoveFirstTwoElementsFrom(h_p)$
        $h_p := h_p + p[i-1] + p[i]$
        if ($s \neq p[i]$) then
            $dism := \mu_{DIT(\mathbf{PG})}(h_s, h_p)$
            if ($dism$ is "high") then
                return "the total reconstruction is necessary"
            end
            if ($dism$ is not "low")) then
                $p' := Reconstruction(h_s, p, i, d_r, l_p)$
                if ($p'$ is empty) then
                    return "the total reconstruction is necessary"
                end
                $p := p'$
            end
        end
        $i := i + 2$ // Go to the next state
    end
end
```

The computational complexity of the Algorithm 7.5 depends linearly on the complexity of algorithm $EEFS$. However, in practice the application of this algorithm may significantly accelerate the execution of plans requiring approximation instead of a total reconstruction. Only a partial reconstruction of the plan is performed whose degree of computational difficulty is much smaller than

Algorithm 7.5. The partial reconstruction of a plan (*Reconstruction*)

Input:
- path $h = (h_1, ..., h_k)$ in the planning graph **PG** representing history of a given complex object, that is finished by an initial state to start of reconstruction,
- plan p_1 generated for a given complex object before the reconstruction,
- position *pos* of initial state of the reconstruction in the plan p_1,
- maximal depth d_r of the plan reconstruction,
- maximal length l_p of a repair plan during the reconstruction.

Output: The plan p_1 after reconstruction

```
1  Procedure Reconstruction($h, p_1, pos, d_r, l_p$)
2  begin
3      $j := pos$
4      $s := GetLastElementFrom(h)$
5      while $j \leq pos + 2 \cdot (d_r - 1)$ do
6          $p_2 := EEFS(h, p_1[j], l_p)$
7          if ($p_2$ is not empty) then
8              $p_3 := Subpath(p_1, 1, pos-1) + p_2 + Subpath(p_1, j+1, length(p_1))$
9              return $p_3$
10         end
11         $j := j + 2$
12     end
13     return "empty plan"
14 end
```

the degree of difficulty of a total reconstruction (because of a smaller size of the problem of the partial reconstruction in relation to the size of the total reconstruction problem).

In practical applications there often occurs a situation that the reconstructed plan must have the same length as the original one. It happens that way when, *e.g.*, a plan proposed by the expert, which is to be executed over the established number of time units (*e.g.*, minutes, hours, days), must be reconstructed. A question arises, if in such a situation the duration time of partial reconstruction executed with the help of the algorithm $EEFS$ is in fact always shorter than the time of total reconstruction which is executed with the same algorithm $EEFS$? After all, the increase of the maximum reconstruction depth causes that in case of not finding the return state, the procedure $EEFS$ must be executed several times for the next reconstruction depths, that is, for $d_r = 1, 2, ..., n_p - 1$, where n_p is the length of reconstructed plan. If the reconstruction algorithm are used, assuming that the maximum reconstruction depth $d_r = n_p$ or even $d_r > n_p$, then obviously such a reconstruction would be more time costly for many plans than total reconstruction. Even in the case when $d_r < n_p$ it could seem that partial reconstruction executed using algorithm $EEFS$ may be for certain plans more time costly than total reconstruction. However, this simple intuition is contradicted by the proposition presented below.

Proposition. *Let us assume that:*

- *p is a plan of n_p length which requires a reconstruction, where $n_p > 0$,*
- *AP is an automatic planning algorithm that its time cost is expressed using function $T(n) = C \cdot m^n$, where C is a constant, m is the maximum number of actions which are considered during the planning of an action in a given state and n is the maximum length of the plan under construction (the time complexity of AP is very similar to time complexity of algorithm EEFS).*

If a reconstruction reconstructs a plan of the same length as before this reconstruction, then for any maximum reconstruction depth $d_r < n_p$ the partial reconstruction executed with algorithm AP takes less time than total reconstruction.

Proof. The cost of total reconstruction of the plan p is $T(n_p)$, whereas the cost of partial reconstruction, with the maximum reconstruction depth d_r, is $T(1) + T(2) + ... + T(d_r)$. Therefore, partial reconstruction works faster than the total one when:

$$T(1) + T(2) + ... + T(d_r) < T(n_p)$$

that is:

$$C \cdot m^1 + C \cdot m^2 + ... + C \cdot m^{d_r} < C \cdot m^{n_p} \tag{7}$$

Hence:

$$m^{n_p+1} - m^{n_p} - m^{d_r+1} + m > 0 \tag{8}$$

It is sufficient to show that (8) is satisfied for any $m > 1$ and $0 < d_r < n_p$.

Starting from the left side of (8) we obtain:

$$m^{n_p+1} - m^{n_p} - m^{d_r+1} + m > m^{n_p+1} - m^{n_p} - m^{n_p-1+1} + m =$$

$$m^{n_p+1} - 2 \cdot m^{n_p} + m = m^{n_p}(m-2) + m \geq m^{n_p}(2-2) + m = m > 0$$

This completes the proof. □

On the basis of the above proposition one may state that partial reconstruction is always more effective than the total one, regardless of the plan length, maximum number of actions which may be performed in a given state and the maximum reconstruction depth. It must be stressed here, however, that partial reconstruction cannot always reconstruct a plan. In such a situation, the use of total reconstruction is the only option.

7.10 Automated Planning for Structured Complex Objects

In planning the behavior of structured objects, an effective planning of the behaviors of all objects which are parts of these objects at the same time is not possible. Therefore, in such cases the behavior of all objects which are parts of a structured object is planned separately. However, this approach to planning of the behavior for a structured object requires a certain synchronization of the plans constructed for individual parts in such a way that these plans would not contradict each other and even complement each other in order to plan the best behavior for a structured object. For example, the treatment of illness A which is the resultant of two illnesses B and C requires such illnesses B and C treatment that the treatments of both illnesses would not be contradictory to each other, but even support and complement each other. For example, it may happen that in treating illness B a certain medicine M_1 may be used which is usually an appropriate medicine but it may be applied only when illness C does not occur. Hence, the synchronization of both illnesses' treatment should exclude the application of medicine M_1. In a different situation it may happen that as a result of application of medicine M_2 for illness C the treatment of illness B is safer, for instead of giving a certain strong medicine M_3, which has negative side effects, it is enough to give a safer medicine M_4 which leads to the same improvement in the patient's condition as in the case of giving medicine M_3.

In a few next subsections we present a generalization of the method for automated planning described in previous subsection for structured objects.

It is worth noticing that in literature one may observe the increase of interest in learning methods of common behaviors of structured objects. This issue is known under the term of learning communication protocols, cooperation and competition (see, *e.g.*, [351, 352]).

7.11 Planning Graphs for Structured Objects

In this paper, the elementary concept allowing planning the behavior of structured objects is the planning graph for structured objects.

Definition 65 (*A planning graph for structured objects*)**.** *A planning graph for structured objects of a fixed type T is a triple $\mathcal{PG} = (\mathcal{S}, \mathcal{A}, \mathcal{E})$ such that $(\mathcal{S}, \mathcal{A}, \mathcal{E})$ is the planning graph, where:*

- *elements of the set $\mathcal{S}$ are called meta states and they represent states of structured objects of the type T,*
- *elements of the set $\mathcal{A}$ are called meta actions and they represent actions for structured objects of the type T,*
- *elements of sets $PATH(\mathcal{PG}, k)$ (where $k > 1$) and $PATH(\mathcal{PG})$ are called meta paths,*
- *elements of sets $PLAN(\mathcal{PG}, k)$ (where $k > 1$) and $PLAN(\mathcal{PG})$ are called meta plans.*

In Fig. 46, we present an exemplary planning graph for a structured object, that is a group of four diseases: sepsis, Ureaplasma, RDS and PDA, related to the planning of the treatment of the infant during the respiratory failure (see Appendix B). This graph was created on the basis of observation of medical data sets (see Section 7.21) and with support of human experts.

As we see, there are two kinds of nodes in the planning graph for structured object, namely, *meta states nodes* (denoted by ovals) that represent the current

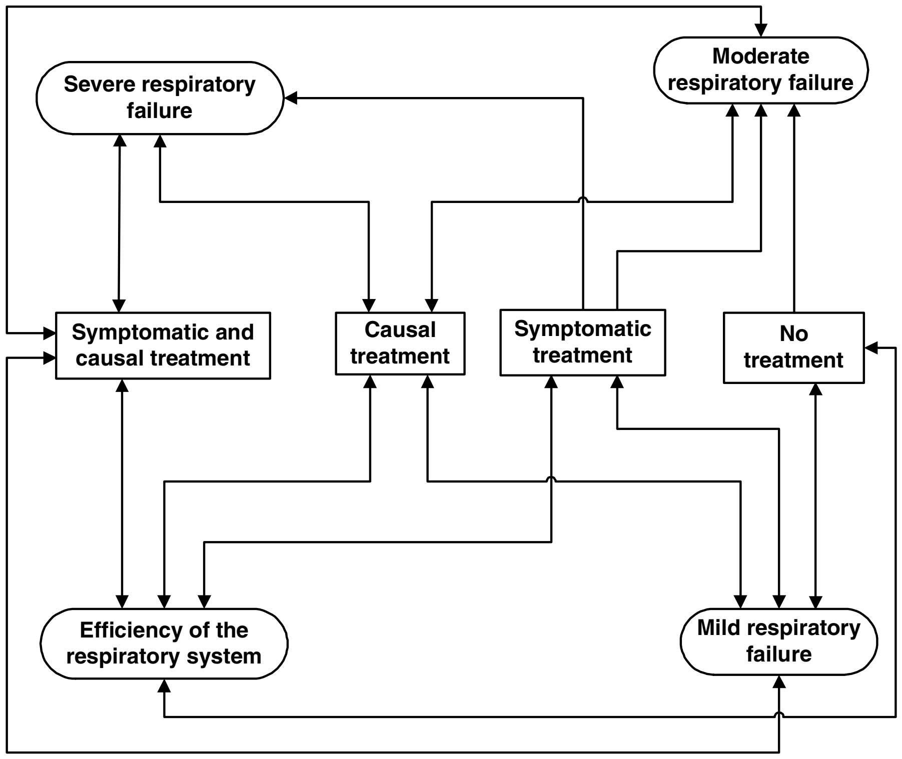

Severe respiratory failure
Moderate respiratory failure
Symptomatic and causal treatment
Causal treatment
Symptomatic treatment
No treatment
Efficiency of the respiratory system
Mild respiratory failure

Fig. 46. A planning graph for the treatment of infants during the respiratory failure

state of a structured object specified as complex concepts by a human expert in natural language, and *meta action nodes* (denoted by rectangles) that represent actions defined for structured objects.

The major difference between the planning graph for the unstructured complex object and the planning graph for the structured object is that in the last one instead of actions performed at a single time point meta-actions occur which are performed over a longer period of time, that is, a time window.

Similarly to the case of unstructured complex objects the problem of planning for the structured object consists in constructing such a plan in planning graph $\mathcal{PG}$ that leads the structured object from the initial state to the expected target state.

7.12 The Identification of the Meta State

At the beginning of planning for a structured object, we identify the current meta state of this object. Any meta state node from a planning graph for structured objects can be treated as a complex spatio-temporal concept that is specified by a human expert in natural language. Such concepts can be approximated by classifiers using data sets and domain knowledge accumulated for a given complex dynamical system. Similarly to states from the planning graph for unstructured complex objects, any state from the planning graph for structured objects can be approximated as a temporal concept for unstructured object (see Section 7.5). However, the state from the planning graph for structured objects can be also treated as the temporal concept for structured objects. Therefore, in this case the method of approximation from Section 6.22 can be used instead of the method from Section 6.9. As a result, it is possible to recognize the initial state at the beginning of planning for a particular structured object.

7.13 Planning of a Meta Action

Similarly to the single complex object, during planning for some structured object the path in the planning graph from the initial node-state to the target node-state should be found. At the beginning of planning for a structured object, we identify the current state of this object. As mentioned earlier, this can be done by classifiers that have been constructed for all states from the planning graph. Next, we plan a sequence of meta actions for transforming a structured object from the current meta state to the target meta state (more expected, safer or more comfortable). For example, in the case of the treatment of infants with respiratory failure, if the infant is suffering from severe respiratory failure, we try to change the patient status using some methods of treatment to change its status to moderate or mild respiratory failure (see Fig. 46). However, any meta action from such constructed path should be checked on the lower level, i.e., on the level of any part of the structured object separately, if such action can be realized in practice in case of particular part of this structured object. In other words, it means that for any part of the structured object the sequence of action should be planed in order to obtain meta-action on the level of the structured object.

The plan of execution of a single meta-action, which consists of short plans which execute this meta-action on the levels of individual parts of the structured object, is called *a g-plan.*

Definition 66 (*A g-plan*)**.** *Let us assume that:*

- $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t}, a_c)$ *is a temporal information system with actions,*
- *T is a type of structured objects, where objects of this type are composed of l parts of object of types T_1,...,T_l,*
- $\mathbf{PGF} = \{\mathbf{PG}_1, ..., \mathbf{PG}_l\}$ *is a family of planning graphs, where* $\mathbf{PG}_i = (S_i, A_i, E_i)$ *is a planning graph for complex objects of type T_i, for $i = 1, ..., l$,*
- *k is a fixed plan length in planning graphs from the family* $\mathbf{PGF}$.

1. *A g-plan with length k for the family of planing graphs* $\mathbf{PGF}$ *is a family of plans $\{p_1, ..., p_l\}$ (assigned to be executed for all parts of the established structured object) such that $p_i \in PLAN(\mathbf{PG}_i, k)$ for $i = 1, ...l$. The set of all g-plans with length k for the family of planing graphs* $\mathbf{PGF}$ *is denoted by* $GPLAN(\mathbf{PGF}, k)$.

2. *Any g-plan $\{p_1, ..., p_l\} \in GPLAN(\mathbf{PGF}, k)$ is called a g-plan occurring in system $\mathbf{T}$, if $p_i \in DPLAN(\mathbf{T}, \mathbf{PG}_i, k)$, for $i = 1, ..., l$ and in system $\mathbf{T}$ there exists such a family of time windows $\{W_1, ..., W_l\} \subseteq TW(\mathbf{T}, k)$ that the following conditions are satisfied:*
   - $\forall_{W_i \in \{W_1,...,W_l\}} W_i \in TW(\mathbf{T}, p_i)$,
   - $\forall_{j \in \{1,...,k\}}\ a_t(W_1[j]) = a_t(W_2[j]) = ... = a_t(W_l[j])$.
3. *A set of all g-plans occurring in the system $\mathbf{T}$ with length k and constructed for the family of planing graphs $\mathbf{PGF}$, is denoted by $DGPLAN(\mathbf{T}, \mathbf{PGF}, k)$.*

The g-plan is, thus, a family of plans assigned to be executed for all parts of the established structured object. The g-plan occurs in the temporal information system with actions if its performance is observed in the data.

Let us notice that determining the $DGPLAN(\mathbf{T}, \mathbf{PGF}, k)$ requires not only determining sets $DPLAN$ for all parts of the structured object but also synchronizing them in time. There arises a problem of isolating structured objects. If we assume, however, that the structured objects are created with the help of the sweeping algorithm around parts of structured objects (see Section 6.13), then the problem of determining the set $DGPLAN$ is significantly simpler.

In practise, all constructed plans for objects (parts) belonging to a given structured object should be compatible. Therefore, during planning a meta action for a structured object, we use a special tool for verifying the compatibility of plans generated for all members of a structured object. This verification can be performed by using some special decision rules that we call *elimination rules*. Such rules make it possible to eliminate combination of plans that are not compatible relative to domain knowledge. This is possible because elimination rules describe all important dependencies between plans that are joined together. If any combination of plans is not consistent with any elimination rule, then it is eliminated. A set of elimination rules can be specified by human experts or can be computed from data sets. In both of these cases, we need a set of attributes (features) defined for a single plan that are used for explaining elimination rules. Such attributes are specified by human experts on the basis of domain knowledge and they describe some important features of the plan (generated for some part of structured object) with respect to proper joining a plan with plans generated for other parts of structured object.

These features are used as a set of attributes in the special table that we call *an elimination table*. Any row of an elimination table represents information about features of plans assigned for structured objects from the training data.

Definition 67 (*An elimination table*)**.** *Let us assume that:*

- $\mathbf{T}$ *is a temporal information system with actions,*
- T *is a type of structured objects, where objects of this type are composed of* l *parts of object of types* T_1,...,T_l,
- $\mathbf{PGF} = \{\mathbf{PG}_1, ..., \mathbf{PG}_l\}$ *is a family of planning graphs, where* $\mathbf{PG}_i = (S_i, A_i, E_i)$ *is a planning graph for complex objects of type* T_i, *for* $i = 1, ..., l$,
- k *is a fixed plan length in planning graphs from the family* $\mathbf{PGF}$,

- $\Phi_i = \{\phi_i^1, ..., \phi_i^{m_i}\} \subseteq FPPG(\mathbf{PG}_i)$ *is a family of formulas defined by experts, for* $i = 1, ..., l$*,*
- $\Phi = \Phi_1 \cup \ ... \ \cup \ \Phi_l$*,*
- $\mathcal{P}_{GP} = (U, \Phi, \models_{GP})$ *is a property system, where*

$$U = DGPLAN(\mathbf{T}, \mathbf{PGF}, k) \text{ and}$$

the satisfiability relation $\models_{GP} \subseteq U \times \Phi$ *is defined in the following way:*

$$\forall gp = \{p_1, ..., p_l\} \in U \text{ and } \phi \in \Phi: \ gp \models_{GP} \phi \ \Leftrightarrow$$

$$p_i \models_{FPPG(\mathbf{PG}_i)} \phi, \quad \text{for } i \in \{1, .., l\} \text{ such that } \phi \in \Phi_i.$$

The information system defined by the property system $\mathcal{P}_{GP}$ *is called an elimination table of g-plans from the set* $GPLAN(\mathbf{PGF}, k)$*.*

It is easy to notice that if the set $DGPLAN(\mathbf{T}, \mathbf{PGF}, k)$ has already been determined and the examination of formula satisfiability may be executed over constant time, then the algorithm which determines the elimination table works over time of order $O(n \cdot m)$, where:

$$n = card(DGPLAN(\mathbf{T}, \mathbf{PGF}, k)) \text{ and } m = card(\Phi_1 \cup ... \cup \Phi_l).$$

Example 37. The respiratory failure may be treated as a result of four following diseases: RDS, PDA, sepsis and Ureaplasma. Therefore, treating respiratory failure requires simultaneous treatment of all of these diseases. This means that the treatment plan of respiratory failure comes into existence by joining the treatment plans for diseases RDS, PDA, sepsis and Ureaplasma, and at the same time the synchronization of the plans is very important. In this paper, one of the synchronizing tools for this type of plans is the elimination table. In constructing the elimination table for treatment of respiratory failure, patterns describing the properties of the joint plans are needed. Moreover, planning graphs for all four diseases are necessary. In Fig. 44 the planning graph for RDS treatment is shown, whereas in Example 36 we showed how the features of RDS treatment plans may be defined. In a very similar way the features of treatment plans for PDA, sepsis and Ureaplasma diseases may be defined. However, in this paper we do not present the planning graphs for treating these diseases. The reason for this is a high degree of complexity of these graphs in terms of medical knowledge (particularly in the case of treating disease sepsis). Therefore, we also cannot give examples of specific features which may be used to describe the treatment plans for diseases: PDA, sepsis and Ureaplasma (as we did in the case of RDS treatment) (see Example 36). □

On the basis of the elimination table a set of elimination rules can be computed that can be used to eliminate inappropriate plan arrangements for individual parts of the structured object.

Definition 68 (*An elimination rule*). *Let us assume that:*

- $\mathbf{T}$ *is a temporal information system with actions,*
- T *is a type of structured objects, where objects of this type are composed of* l *parts of object of types* T_1,...,T_l,
- $\mathbf{PGF} = \{\mathbf{PG}_1, ..., \mathbf{PG}_l\}$ *is a family of planning graphs, where* $\mathbf{PG}_i = (S_i, A_i, E_i)$ *is a planning graph for complex objects of type* T_i, *for* $i = 1, ..., l$,
- k *is a fixed plan length in planning graphs from the family* $\mathbf{PGF}$,
- $\mathbf{ET}(\mathbf{T}, \mathbf{PGF}, k) = (U, A)$ *is an elimination table.*

1. *If* $a \in A$ *then any decision rule with minimal number of descriptors (see Section 2.3 and Section 2.4) computed for a decision table* $(U, A \setminus \{a\}, a)$ *is called an elimination rule for the elimination table* $\mathbf{ET}(\mathbf{T}, \mathbf{PGF}, k)$.
2. *A set of all elimination rules for the elimination table* $\mathbf{ET}(\mathbf{T}, \mathbf{PGF}, k)$ *is denoted by* $\mathbf{ERUL}(\mathbf{T}, \mathbf{PGF}, k)$.
3. *If* $r \in \mathbf{ERUL}(\mathbf{T}, \mathbf{PGF}, k)$, *then an object* $u \in U$ *is eliminated by the elimination rule* r *iff* u *matches the predecessor of* r *and does not match the successor of the rule* r.

Algorithm 7.6. Generation of elimination rules

```
Input:
  – temporal information system with actions $\mathbf{T}$,
  – type $T$ of structured objects, where objects of this type are composed
    of $l$ parts of object of types $T_1$,...,$T_l$,
  – family of planning graphs $\mathbf{PGF} = \{\mathbf{PG}_1, ..., \mathbf{PG}_l\}$, where
    $\mathbf{PG}_i = (S_i, A_i, E_i)$
    is a planning graph for complex objects of type $T_i$, for $i = 1, ..., l$,
  – fixed plan length $k$ from planning graphs from the family $\mathbf{PGF}$,
  – elimination table $\mathbf{ET}(\mathbf{T}, \mathbf{PGF}, k) = (U, A)$ such that $A = \{a_1, ...., a_m\}$,
  – minimal support $t_s$ of useful elimination rules.
Output: The set of elimination rules computed for the table
        $\mathbf{ET}(\mathbf{T}, \mathbf{PGF}, k)$
1  begin
2      Create empty set of rules $ERUL$
3      for any $a \in A$ do
4          Create a decision table $\mathbf{ET}_a = (U_a, A_a, d)$ such that, $U_a = U$,
           $A_a = A \setminus \{a\}$ and $d = a$
5          Generate a set $RUL(\mathbf{ET}_a)$ of decision rules with minimal number
           of descriptors (see Section 2.3 and Section 2.4) for the table $\mathbf{ET}_a$
6          Add rules from the set $RUL(\mathbf{ET}_a)$ to the set $ERUL$
7      end
8      Remove from the set $ERUL$ all rules with support less than $t_s$
9      return $ERUL$
10 end
```

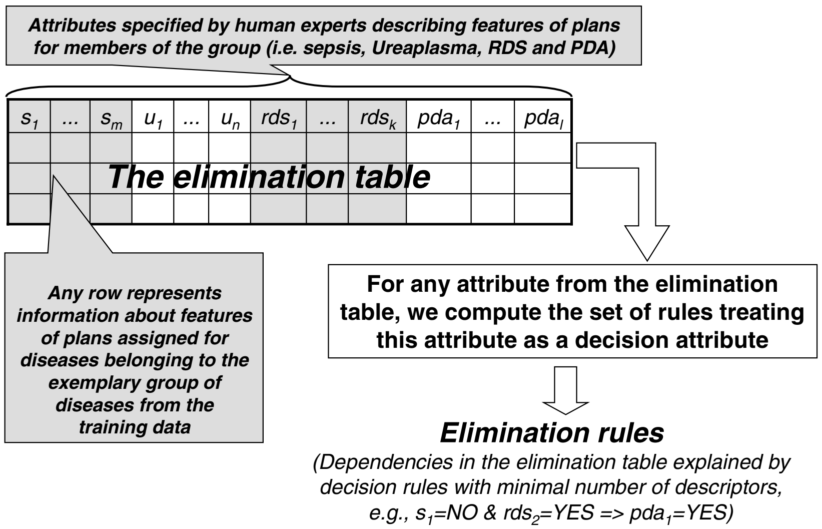

Attributes specified by human experts describing features of plans for members of the group (i.e. sepsis, Ureaplasma, RDS and PDA)
s_1 ... s_m u_1 ... u_n rds_1 ... rds_k pda_1 ... pda_l
The elimination table
Any row represents information about features of plans assigned for diseases belonging to the exemplary group of diseases from the training data
For any attribute from the elimination table, we compute the set of rules treating this attribute as a decision attribute
Elimination rules
(Dependencies in the elimination table explained by decision rules with minimal number of descriptors, e.g., s_1=NO & rds_2=YES => pda_1=YES)

Fig. 47. The scheme of construction of elimination rules for group of four diseases: sepsis, Ureaplasma, RDS and PDA

So, the set of elimination rules can be used as a filter of inconsistent combinations of plans generated for members of groups. Any combination of plans is eliminated when there exists an elimination rule that is not supported by features of a combination while the combination matches a predecessor of this rule. In other words, a combination of plans is eliminated when the combination matches to the predecessor of some elimination rule and does not match the successor of a rule.

We propose the following method of calculation the set of elimination rules on the basis of the elimination table (see Algorithm 7.6).

As we see in the Algorithm 7.6, for any attribute from the elimination table, we compute the set of rules with minimal number o descriptors (see Section 2.3 and Section 2.4) treating this attribute as a decision attribute. In this way, we obtain a set of dependencies in the elimination table explained by decision rules. In practice, it is necessary to filter elimination rules to remove the rules with low support because such rules can be too strongly matched to the training data.

Fig. 47 shows the scheme of elimination rules of not-acceptable g-plans constructed in the case of the treatment of respiratory failure, which is a result of the four following diseases: sepsis, Ureaplasma, RDS and PDA.

On the basis of the set of elimination rules *an elimination classifier* may be constructed that enable elimination of inappropriate plan arrangements for individual parts of the structured object.

Definition 69 (*An elimination classifier*)**.** *Let us assume that:*

- $\mathbf{T}$ *is a temporal information system with actions,*
- T *is a type of structured objects, where objects of this type are composed of* l *parts of object of types* T_1,...,T_l,

- $\mathbf{PGF} = \{\mathbf{PG}_1, ..., \mathbf{PG}_l\}$ *is a family of planning graphs, where* $\mathbf{PG}_i = (S_i, A_i, E_i)$ *is a planning graph for complex objects of type* T_i*, for* $i = 1, ..., l$*,*
- k *is a fixed plan length in planning graphs from the family* $\mathbf{PGF}$*,*
- $\mathbf{ET}(\mathbf{T}, \mathbf{PGF}, k) = (U, A)$ *is an elimination table,*
- $\mathbf{ERUL}(\mathbf{T}, \mathbf{PGF}, k)$ *is a set of all elimination rules of g-plans from the set* $GPLAN(\mathbf{PGF}, k)$*.*

An elimination classifier based on the set $\mathbf{ERUL}(\mathbf{T}, \mathbf{PGF}, k)$ *of all elimination rules (or on some subset of this set) is a classifier denoted in general by* $\mu_{\mathbf{ET}(\mathbf{T}, \mathbf{PGF}, k)}$ *and classifying g-plans in the following way:*

$$\forall u \in U : \mu_{\mathbf{ET}(\mathbf{T}, \mathbf{PGF}, k)}(u) = \begin{cases} false & when\ \exists_{r \in ERUL}\ u\ is\ eliminated\ by\ r \\ true & otherwise. \end{cases}.$$

If the combination of plans for parts of the structured object is consistent (it was not eliminated by elimination rules), we should check if the execution of this combination allows us to realize the expected meta action from the level of structured objects. This can be done by a special classifier constructed for a table called a *meta action table.* The structure of a meta action table is similar to the structure of an elimination table, i.e., attributes are defined by human experts, where rows represent information about features of plans assigned for parts of exemplary structured objects from the training data. In addition, we add to this table a decision attribute. Values of such decision attributes represent names of meta actions which are realized as an effect of the execution of plans described in the current row of a training table.

Definition 70 (*A meta action table*). *Let us assume that:*

- $\mathbf{T}$ *is a temporal information system with actions,*
- T *is a type of structured objects, where objects of this type are composed of* l *parts of object of types* T_1,...,T_l*,*
- $\mathcal{PG} = (\mathcal{S}, \mathcal{A}, \mathcal{E})$ *is a planning graph for structured objects of the type* T*,*
- $\mathbf{PGF} = \{\mathbf{PG}_1, ..., \mathbf{PG}_l\}$ *is a family of planning graphs, where* $\mathbf{PG}_i = (S_i, A_i, E_i)$ *is a planning graph for complex objects of type* T_i*, for* $i = 1, ..., l$*,*
- k *is a fixed plan length in planning graphs from the family* $\mathbf{PGF}$*,*
- $\Phi_i = \{\phi_i^1, ..., \phi_i^{m_i}\} \subseteq FPPG(\mathbf{PG}_i)$ *is a defined by experts family of formulas, for* $i = 1, ..., l$*,*
- $\Phi = \Phi_1 \cup\ ...\ \cup \Phi_l$*,*
- $\mathcal{P}_{GP} = (U, \Phi, \models_{GP})$ *is a property system, where*

$$U = DGPLAN(\mathbf{T}, \mathbf{PGF}, k)\ and$$

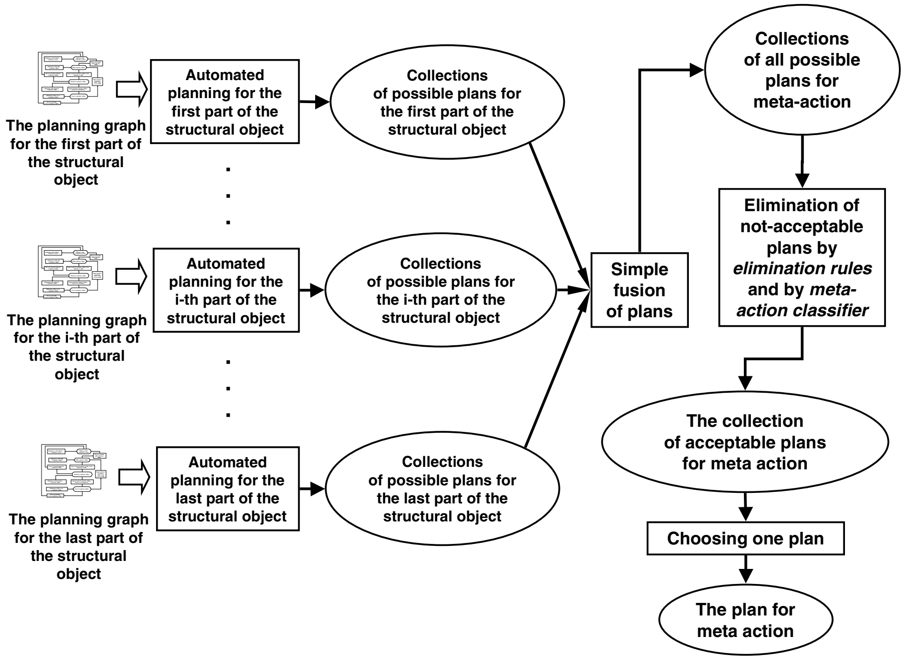
the satisfiability relation $\models_{GP} \subseteq U \times \Phi$ *is defined in the following way:*

$$\forall gp = \{p_1, ..., p_l\} \in U\ and\ \ \phi \in \Phi :\ gp \models_{GP} \phi\ \Leftrightarrow$$

$$p_i \models_{FPPG(\mathbf{PG}_i)} \phi,\ \ for\ i \in \{1, .., l\}\ such\ that\ \phi \in \Phi_i.$$

1. *A meta action table of g-plans for structured objects is a decision table* $\mathbf{MAT}(\mathbf{T}, \mathbf{PGF}, k) = (U, A, d)$*, where:*
   - (U, A) *is an information system defined be the property system* $\mathcal{P}_{GP}$*,*
   - d *is a decision attribute that for any g-plan from the set* U*, represents a meta action corresponding to execution of this g-plan.*
2. *If* $\mathbf{MAT}(\mathbf{T}, \mathbf{PGF}, k)$ *is the meta action table, then any classifier computed for the table* $\mathbf{MAT}(\mathbf{T}, \mathbf{PGF}, k)$ *is called a meta action classifier and is denoted by* $\mu_{\mathbf{MAT}(\mathbf{T}, \mathbf{PGF}, k)}$.

The classifier computed for an action table makes it possible to predict the name of a meta action for a given combination of plans from the level of parts of a structured object. The last step is the selection of combinations of plans that makes it possible to obtain a target meta action with respect to a structured object (see Fig. 48).



The planning graph for the first part of the structural object
Automated planning for the first part of the structural object
Collections of possible plans for the first part of the structural object
The planning graph for the i-th part of the structural object
Automated planning for the i-th part of the structural object
Collections of possible plans for the i-th part of the structural object
The planning graph for the last part of the structural object
Automated planning for the last part of the structural object
Collections of possible plans for the last part of the structural object
Simple fusion of plans
Collections of all possible plans for meta-action
Elimination of not-acceptable plans by *elimination rules* and by *meta-action classifier*
The collection of acceptable plans for meta action
Choosing one plan
The plan for meta action

Fig. 48. The scheme of meta action planning

Example 38. The treatment of respiratory failure requires simultaneous treatment of RDS, PDA, sepsis and Ureaplasma. Therefore, the treatment plan for respiratory failure comes to existence by joining the treatment plans for RDS, PDA, sepsis and Ureaplasma, and at the same time the synchronization of those plans is very important. The first tool to synchronize these types of plans is the elimination classifier generated for the elimination table. The second tool, however, is the meta action classifier generated for the meta action table. Similarly to the case of the elimination table, also in constructing the meta action table, patterns describing the properties of the joint treatment plans for RDS, PDA, sepsis and Ureaplasma are needed. These patterns are very similar as in the case of the patterns used to construct the elimination table (see Example 37). □

It was mentioned in Section 7.3 that the resolving classifier used for generation of a next action during the planning for a single object, gives us the list of actions (and states after usage of action) with their weights in descending order. This makes it possible to generate many alternative plans for any single object and many alternative combinations of plans for a structured object. Therefore, the chance of finding an expected combination of plans from a lower level to realize a given meta action (from the higher level) is relatively high.

After planning the selected meta action from the path of actions from the planning graph (for a structured object), the system begins the planning of the next meta action from this path. The planning is stopped, when the planning of the last meta action from this path is finished.

7.14 Data Structures, Algorithms and Numerical Constants Concerning the Planning for Structured Objects

In the next subsections, we present several algorithms which are needed to plan the behavior of structured objects. Therefore, in this subsection we formulate the problem of behavior planning of such objects and we mention elementary data structures, algorithms, and numerical constants which we use in the described algorithms. This allows avoiding repetitive descriptions of elements of such a kind.

When we speak about the structured object's behavior planning, we always mean the behavior planning of a certain structured object O of the established type T which consists of parts $O_1, ..., O_l$ which are respectively the objects of types $T_1, ..., T_l$. In practical applications, objects $O_1, ..., O_l$ are often unstructured objects, however, they also may be structured objects of lesser complexity than object O. Hence, the methodology of structured object behavior planning described here is of a hierarchical character.

In this paper, in automatic planning of complex object behavior we use data sets represented by the temporal information system with actions $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t}, a_c)$. We also use the following data structures, algorithms, and numerical constants:

- a planning graph $\mathcal{PG} = (\mathcal{S}, \mathcal{A}, \mathcal{E})$ for structured objects of a fixed type T,
- a fixed plan length K in planning graphs $\mathcal{PG}$,
- a classifier $\mu_{\mathbf{RC}(\mathcal{PG},S,K)}$, for all $S \in \mathcal{S}$,
- a classifier of dissimilarity $\mu_{DIT(\mathcal{PG})}$ between meta states from the planning graph $\mathcal{PG}$,
- a limitation *ActionLimit* of the number of meta actions which may be performed in a given state $S \in \mathcal{S}$,
- a maximal length L_p of generated plan for structured objects,
- a maximal depth D_r of a plan reconstruction for structured objects,
- a maximal length L_{rp} of a repair plan for structured objects (during reconstruction),

- a family of planning graphs $\mathbf{PGF} = \{\mathbf{PG}_1, ..., \mathbf{PG}_l\}$, where $\mathbf{PG}_i = (S_i, A_i, E_i)$ is a planning graph for complex objects of type T_i, for $i = 1, ..., l$,
- a fixed length o plans k in planning graphs from the family $\mathbf{PGF}$,
- a classifier of dissimilarity $\mu_{DIT(\mathbf{PG}_i)}$ between states from the planning graph $\mathbf{PG}_i$, for $i = 1, ..., l$,
- a length l_{gp} of generated g-plan constructed for execution of any meta action from the set $\mathcal{A}$,
- an elimination classifier $\mu_{\mathbf{ET}(\mathbf{T},\mathbf{PGF},k)}$ (see Definition 69),
- a meta action classifier $\mu_{\mathbf{MAT}(\mathbf{T},\mathbf{PGF},k)}$ (see Definition 70),
- a maximal depth d of reconstruction of g-plans,
- a maximal length l_{rp} of a repair of g-plan.

The mentioned above data structures, algorithms, and numerical constants are used in algorithms presented in further subsections.

7.15 Algorithms of the Meta Action Planning

The basic method of automated planning for the structured object is the method of planning of the meta-action. The planning of the established meta-action requires constructing such a g-plan that has a required length, it is compatible with domain knowledge and its execution corresponds to the performance of the established meta-action.

We present an example of an automated planning algorithm for a meta-action which has been used in our experiments (see Algorithm 7.7).

The procedure *SearchGPlanMA* (see Algorithm 7.7) constructs a g-plan for the structured object which executes the required meta-action. First, the sets of plans are generated for all the parts of the structured object separately. Then, the Cartesian product $PCList$ of these sets is created, whose elements are candidates for the g-plan that is being searched for. Finally, this product is overviewed until finding such a g-plan that is not eliminated by classifier $\mu_{\mathbf{ET}(\mathbf{T},\mathbf{PGF},k)}$ and performs the required meta-action on the structured object according to classifier $\mu_{\mathbf{MAT}(\mathbf{T},\mathbf{PGF},k)}$. If during the overview of the set $PCList$ such a g-plan occurs, then the execution of the algorithm is ended and exactly this g-plan is returned as a solution. If during the overview of the $PCList$ set the algorithm does not encounter such a g-plan, then an empty g-plan is returned which means that the algorithm has not found the solution. However, before the overview of the set $PCList$ takes place, it is sorted. The sorting is necessary for the first g-plan that is found which executes the required meta-action is the most recommended in terms of all parts of the structured object for which the plans are constructed. An example of such sorting g-plans may be the sorting in relation to the weight $GPlanWeight$ which is defined for a given g-plan $gp = (p_1, ..., p_l)$, where $p_i \in PList_i$ for $i = 1, ..., l$, in the following way:

$$GPlanWeight(gp) = \frac{1}{\sum_{i=1}^{l} \text{ Index of } p_i \text{ in the list } PList_i}.$$

The time complexity of the Algorithm 7.7 depends greatly on the size of the set $PCList$. Therefore, the length of the plan lists generated for individual parts

Algorithm 7.7. Searching for g-plan for a meta action ($SearchGPlanMA$)

Input:
- path $h_i \in PLAN(\mathbf{PG}, k)$ in the planning graph $\mathbf{PG}_i$ representing history of the object O_i, that is finished by an initial state to start of automated planning for object O_i, for $i = 1, ..., l$,
- target meta action $ma_t \in \mathcal{A}$,
- length of g-plan l_{gp}.

Output: The g-plan for execution the meta action ma_t

```
1  Procedure $SearchGPlanMA(\{h_1, ..., h_l\}, ma_t, l_{gp})$
2  begin
3      $PList_1 := FEEFS(h_1, l_{gp})$
4      ...
5      $PList_l := FEEFS(h_l, l_{gp})$
6      $PCList := PList_1 \times ... \times PList_l$
7      Sort $PlanCompList$ by established method
8      for $i := 1$ to $length(PCList)$ do
9          $gplan := PCList[i]$
10         if $(\mu_{\mathbf{ET}(\mathbf{T},\mathbf{PGF},k)}(gplan) = true)$ then
11             if $(\mu_{\mathbf{MAT}(\mathbf{T},\mathbf{PGF},k)}(gplan) = ma_t)$ then
12                 return $gplan$
13             end
14         end
15     end
16     return "empty g-plan"
17 end
```

of the structured object should be chosen in such a way that the size of the set $PCList$ would be feasible for searching over this set.

Sometimes, a slightly different algorithm of g-plans searching is necessary. We mean the situation when a g-plan is constructed, that satisfies certain conditions for all parts of the analyzed structured object. In such a situation the Algorithm 7.8 may be used.

As we see, the Algorithm 7.8 constructs such a g-plan for the structured object that ends with specific states for individual parts of the structured object.

7.16 Automated Planning Algorithm for Structured Objects

Now, we may present the planning algorithm for the structured object which uses the *SearchGPlanMA* procedure (see Algorithm 7.9).

The algorithm 7.9 takes into consideration different variants for a meta-action, that may be performed in a given meta-state. However, only those actions are taken into account which algorithm $SearchGPlanMA$ can execute using a g-plan on the level of single parts of the structured object. Similarly to the case of

Algorithm 7.8. Searching for g-plan for states (*SearchGPlan*)

Input:
- path $h_i \in PLAN(\mathbf{PG}, k)$ in the planning graph $\mathbf{PG}_i$ representing history of the object O_i, that is finished by an initial state to start of automated planning for object O_i, for $i = 1, ..., l$,
- target state s_i for the object O_i, for $i = 1, ..., l$,
- length of g-plan l_{gp}.

Output: The g-plan ends by states $s_1, ..., s_l$

```
Procedure $SearchGPlan(\{h_1, ..., h_l\}, \{s_1, ..., s_l\}, l_{gp})$
begin
    $PList_1 := FEEFS(h_1, l_{gp})$
    ...
    $PList_l := FEEFS(h_l, l_{gp})$
    $PCList := PList_1 \times ... \times PList_l$
    Sort $PCList$ by fixed order
    for $i := 1$ to $length(PCList)$ do
        $\{p_1, ..., p_l\} := PCList[i]$
        if $(\mu_{\mathbf{ET}(\mathbf{T},\mathbf{PGF},k)}(gplan) = true)$ then
            if $(p_1[l_p] = s_1)$ and .... and $(p_l[l_p] = s_l)$ then
                return $\{p_1, ..., p_l\}$
            end
        end
    end
    return "empty g-plan"
end
```

Algorithm 7.2, for the regulation of computing time duration, limitation *ActionLimit* is used, that is, limitation of the number of actions which may be performed in a given state. Besides that, classifier $\mu_{\mathbf{RC}(\mathcal{PG},S,K)}$ returns the list of pairs (meta action + meta state) sorted decreasingly in relation to the weights obtained from classification. Hence, the meta-actions are taken into account in the order from the ones most recommended by the classifier $\mu_{\mathbf{RC}(\mathcal{PG},S,K)}$ to the less recommended by this classifier. This way, the algorithm constructs a certain plan tree whose root is the initial meta-state and the leaves are the meta-states after performing the individual variants of the plan. If during construction of this tree the final meta-state occurs, then the performance of the algorithm is ended and as a solution a sequence of meta-states and meta-actions is returned which starts in the tree root and ends in the final meta-state that has been found. If during construction of plan tree the algorithm does not encounter the final meta-state, an empty plan is returned which means that the algorithm has not found the solution.

The analysis of the pessimistic time complexity of Algorithm 7.9 is very similar to the Algorithm 7.2. The only difference is that before applying the meta action

Algorithm 7.9. Exhaustive expert forward search for a structured object

Input:
- path $H = (H_1, ..., H_k) \in PLAN(\mathcal{PG}, k)$ representing history of a given structured object O, that is finished by an initial meta state to start of automated planning,
- family of paths $h_1, ..., h_l$, where h_i is a path from the planning graph $\mathbf{PG}_i$, representing history of the object O_i during execution the last meta action for this object, for $i = 1, ..., l$,
- target meta state $S_t \in \mathcal{S}$,
- a maximal length L_p of plan for structured objects.

Output: The plan P for structured object O ended by the meta state S_t

```
Procedure $EEFSS(H, \{h_1, ..., h_l\}, S_t, L_p)$
begin
    $S := GetLastElementFrom(H)$
    $P :=$ "empty plan"
    $P := P + S$;
    $L := PairList(\mu_{\mathbf{RC}(\mathcal{PG},S,K)}(H))$
    for $i = 1$ to $ActionLimit$ do
        $P_1 := Copy(P)$;
        $gplan := SearchGPlanMA(\{h_1, ..., h_l\}, L[i].action)$
        if ($gplan$ in not empty) then
            $P_1 := P_1 + L[i].action + L[i].state$
            if ($L[i].state = S_t$) then
                return $P_1$
            end
            if ($L_p > 1$) then
                $H_1 := Copy(H)$
                $RemoveFirstTwoElementsFrom(H_1)$
                $H_1 := H_1 + L[i].action + L[i].state$
                $P_2 := EEFSS(H_1, gplan, S_t, L_p - 1)$
                if ($P_2$ is not empty) then
                    return $P + P_2$
                end
            end
        end
    end
    return "empty meta plan"
end
```

the procedure $SearchGPlanMA$ should be executed in order to check how a given meta-action can be executed for the parts of the structured object. If it is even assumed that this checking takes place over constant time, then the pessimistic time complexity of the procedure $EEFSS$ is of order $O(m^n)$ where

n is the length of the constructed plan and m is the limitation of the number of meta-actions which may be performed in a given meta-state (*ActionLimit*). This means that similarly to the case of algorithm $EEFS$ the effective application of the algorithm $EEFSS$ for nontrivially small n and m is practically impossible. However, in practice the planning graphs for the structured object are relatively simple (see Fig. 46) and this algorithm may be used for them.

7.17 Reconstruction of Plan for Structured Objects

The main aim of this section is to present the algorithm of reconstruction of a plan constructed for structured objects. Similarly to the case of the unstructured complex object (see Section 7.9) such a reconstruction may be performed during the execution of the meta plan for the structured object when the initially established plan cannot be continued.

We may present the algorithm simulating the execution of the meta plan which foresees the reconstruction of the plan during its execution (see Algorithm 7.10).

The Algorithm 7.10 simulates the execution of the meta-plan found earlier for the structured object. The simulation is performed based on the procedure *SimulateMA* which on the input takes the history of the current states of all parts of the structured object. The procedure *SimulateMA* returns the g-plan *gp* which has been performed as a result of the simulation of the meta-action. This g-plan may differ from the input g-plan, because during the simulation of the input g-plan a reconstruction on the level of meta-action performance may have occurred (see Algorithm 7.11). Therefore, it may happen that the simulation of the meta-action leads to a different meta-state than the one expected in the original plan. Hence, the procedure *MReconstruction* can be executed in the further process a reconstruction for the structured object (see Algorithm 7.13).

Now, we present the simulation algorithm of meta-action for the structured object with the consideration of reconstruction (see Algorithm 7.11 and Fig. 49).

The simulation is performed based on the procedure *GSimulate* which on the input takes the history of the current states of all parts of the structured object and actions which are to be performed for all these parts. Although it is possible to imagine this type of procedure as a part of the behavior simulator of the complex object (*e.g.*, the traffic simulator, the illness development simulator), in this paper by this procedure we understand the changes in the real system of complex objects which may be triggered by performing specific meta-actions for the structured object.

The Algorithm 7.11 uses the procedure of reconstruction *GReconstruction*. Therefore, we present this procedure as the Algorithm 7.12.

The Algorithm 7.12 tries to find a short repair g-plan not longer than l_{rp} which brings the initial states of reconstruction (the last states in the histories $h_1, ..., h_l$) to the states appearing synchronically in the plans $p_1, ..., p_l$ starting from position *pos* as far as position $pos + 2 \cdot (d_r - 1)$. The maximum depth of reconstruction d_r is then the number of states in the plans $p_1, ..., p_l$ (starting from the states in position *pos*) which the algorithm tries to reach using the repair g-plan. The repair g-plan is searched for by algorithm *SearchGPlan*.

Algorithm 7.10. The simulation of plan execution for a structured object

Input:
- a path $H \in PLAN(\mathcal{PG}, k)$ representing history of a given structured object O, that is finished by an initial meta state to start of a simulation,
- a family of paths $h_1, ..., h_l$, where h_i is a path from the planning graph $\mathbf{PG}_i$, representing history of the object O_i during execution the last meta action for this object (before the simulation), for $i = 1, ..., l$,
- a plan $P \in PLAN(\mathcal{PG}, L_P)$ established for a given complex object O,
- a sequence of g-plans $gp_1, ..., gp_{\frac{L_P-1}{2}}$ implementing meta actions from P.

Output: The plan P executed for the structured object O

```
begin
    if ($length(P) < 3$) return "meta plan P is too short for execution"
    $H_s := Copy(H)$
    $H_p := Copy(H)$
    $i := 1$
    while ($i < length(P)$) do
        $gp := SimulateMA(\{h_1, ..., h_l\}, gp_{\frac{i+1}{2}})$
        if($gp$ is empty) return "start total reconstruction"
        $S := \mu_{\mathbf{MAT}(\mathbf{T},\mathbf{PGF},k)}(gp)$
        $RemoveFirstTwoElementsFrom(H_s)$
        $H_s := H_s + P[i+1] + S$
        $RemoveFirstTwoElementsFrom(H_p)$
        $H_p := H_p + P[i+1] + P[i+2]$
        if ($S \neq P[i+2]$) then
            $dism := \mu_{DIT(\mathcal{PG})}(H_s, H_p)$
            if ($dism$ is "high") then return "start total reconstruction"
            if ($dism$ is not "low") then
                $P' := MReconstruction(H_s, \{h_1, ..., h_l\}, P, i+2)$
                if ($P'$ is empty) then return "start total reconstruction"
                $P := P'$
            end
        end
        $i := i + 2$ // Go to the next meta action from the plan $P$
    end
end
```

Computational complexity of Algorithm 7.12 depends linearly on the complexity of the algorithm *SearchGPlan*. However, in practice using this algorithm may significantly accelerate the execution of plans which require reconstruction because instead of a total reconstruction, only a partial reconstruction is performed.

If the meta state S, achieved as a result of simulation, differs too much from its counterpart in plan P, then the total reconstruction of plan P is performed.

Algorithm 7.11. The simulation of meta-action for the structured object with the consideration of g-plan reconstruction

Input:
- family of paths $h_1, ..., h_l$, where h_i is a path from the planning graph $\mathbf{PG}_i$, representing history of the object O_i during execution the last meta action for this object, for $i = 1, ..., l$,
- meta action $ma \in \mathcal{A}$ and g-plan $gp = (p_1, ..., p_l)$ corresponding to meta action ma, that should be performed during a simulation.

Output: The g-plan executed for the structured object O

```
Procedure $SimulateMA(\{h_1, ..., h_l\}, \{p_1, ..., p_l\})$
begin
    for $j := 1$ to $l$ do
        $hs_j := Copy(h_j)$; $hp_j := Copy(h_j)$
    end
    $i := 1$
    while $(i < l_{gp})$ do
        $\{s_1, ...., s_l\} := GSimulate(\{hs_1, ..., hs_l\}, \{p_1[i+1], ..., p_l[i+1]\})$
        $DISM := false$
        $j := 1$
        while $(j \leq l)$ and $(DISM{=}false)$ do
            $RemoveFirstTwoElementsFrom(hs_j)$
            $hs_j := hs_j + p_j[i+1] + s_j$
            $RemoveFirstTwoElementsFrom(hp_j)$
            $hp_j := hp_j + p_j[i+1] + p_j[i+2]$
            if $(s_j \neq p_j[i+2])$ then
                $dism := \mu_{DIT(\mathbf{PG}_j)}(hs_j, hp_j)$
                if ($dism$ is "$high$") then return "generate new g-plan"
                if ($dism$ is not "$low$") then $DISM := true$
            end
            $j := j + 1$
        end
        if $(DISM{=}true)$ then
            $gplan := GReconstruction(\{hs_1, ..., hs_l\}, \{p_1, ..., p_l\}, j-1)$
            if ($gplan$ is empty ) "generate new g-plan"
            for $j := 1$ to $l$ do $p_j := gplan[j]$
        end
        $i := i + 2$
    end
    return $\{p_1, ..., p_l\}$
end
```

Finally, we present a plan reconstruction algorithm for a structured object on the level of the planning graph for objects of this type (see Algorithm 7.13).

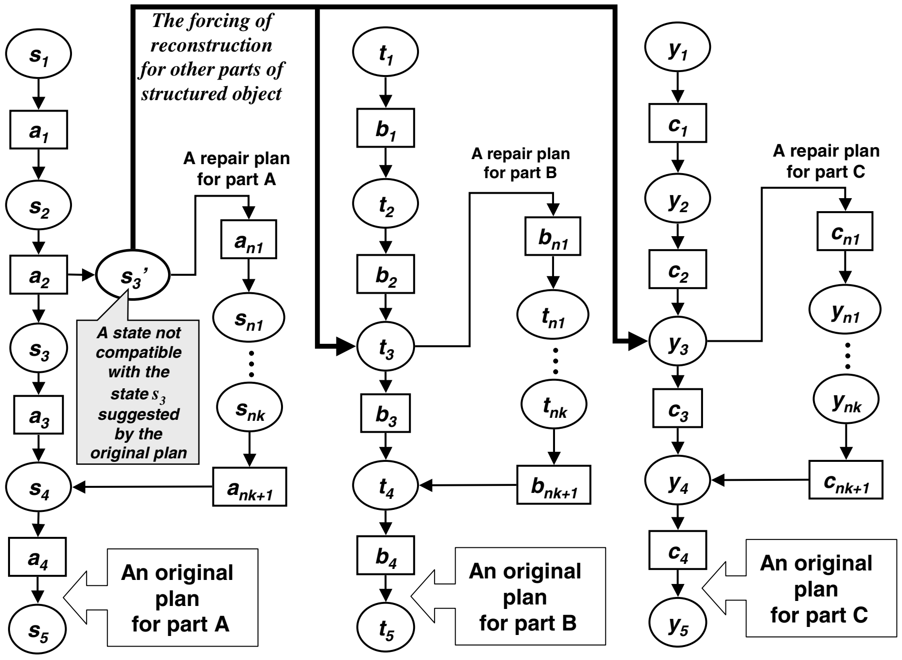

The forcing of reconstruction for other parts of structured object
A repair plan for part A
A repair plan for part B
A repair plan for part C
A state not compatible with the state s_3 suggested by the original plan
An original plan for part A
An original plan for part B
An original plan for part C

Fig. 49. The simulation of meta action execution with reconstruction for a structured object composed of parts: A, B, C

The Algorithm 7.13 tries to find a short repair plan P_2 (not longer than L_{rp}) which brings the initial state of the reconstruction (the last state in history H) to a state appearing in plan P_1, starting from position pos as far as position $pos + 2 \cdot (D_r - 1)$. The maximum depth of reconstruction D_r is, therefore, the number of meta states in plan P_1 (starting from position pos), which the algorithm tries to reach using the repair plan. The repair plan is searched for by algorithm $EEFSS$ although it is possible to use also other planning algorithms.

Computational complexity of the above algorithm depends linearly on complexity of algorithm $EEFSS$. However, in practice using this algorithm may significantly accelerate the execution of meta plans which require reconstruction because instead of a total reconstruction, only a partial reconstruction of the meta-plan is performed whose degree of computational difficulty is much lower than the difficulty level of a total reconstruction (with regard to the smaller size of the problem).

7.18 Estimation of the Similarity between Plans

The problem of inducing classifiers for similarity relations is one of the challenging problems in data mining and knowledge discovery (see, *e.g.*,

Algorithm 7.12. Partially reconstruction of g-plan (*GReconstruction*)

Input:
- family of paths $h_1, ..., h_l$, where h_i is a path from the planning graph $\mathbf{PG}_i$, representing history of the object O_i during execution the last meta action for this object, for $i = 1, ..., l$,
- position *pos* of starting state of reconstruction in plans p_1, ..., p_l.

Output: The reconstructed g-plan $p_1, ..., p_l$

```
Procedure $GReconstruction(\{h_1, ..., h_l\}, \{p_1, ..., p_l\}, pos)$
begin
    $j := pos$
    while $(j \leq pos + 2 \cdot (d_r - 1))$ do
        $\{q_1, ..., q_l\} := SearchGPlan(\{h_1, ..., h_l\}, \{p_1[j], ..., p_l[j]\}, l_{rp})$
        if $(\{q_1, ..., q_l\}$ is not empty$)$ then
            $p_1 := Subpath(p_1, 1, pos-1) + q_1 + Subpath(p_1, j+1, length(p_1))$
            ...
            $p_l := Subpath(p_l, 1, pos-1) + q_l + Subpath(p_l, j+1, length(p_l))$
            return $\{p_1, ..., p_l\}$
        end
        $j := j + 2$
    end
    return "empty g-plan"
end
```

[162, 163, 164, 165, 166, 167, 168, 169, 170, 171]). The existing methods are based on building models for similarity functions using simple strategies for fusion of local similarities. The optimization of the assumed parameterized similarity formula is performed by tuning parameters relative to local similarities and their fusion. For instance, if we want to compare two medical plans of treatments, e.g., one plan generated automatically by our computer system and another one proposed by medical expert, we need a tool to estimate the similarity. This problem can be solved by introducing a function measuring the similarity between medical plans. For example, in the case of our medical data (see Section 7.21), a formula is used to compute a similarity between two plans as the arithmetic mean of similarity between all corresponding pairs of actions (nodes) from both plans, where the similarity for the single corresponding pair of actions is defined by a consistence measure of medicines and medical procedures comprised in these actions. For example, let $M = \{m_1, ..., m_k\}$ be a set consisting of k medicines. Let us assume that actions in medical plans are specified by subsets of M. Hence, any medical plan P determines a sequence of actions $\mathcal{A}(P) = (A_1, ..., A_n)$, where $A_i \subseteq M$ for $i = 1, \ldots, n$ and n is the number of actions in P. In our example, the similarity between plans is defined by a similarity function *Sim* established

Algorithm 7.13. Reconstruction of the plan for structured objects (*MReconstruction*)

Input:
- path H in the planning graph $\mathcal{PG}$ representing history of a given structured complex object O, that is finished by an initial state to start of reconstruction,
- family of paths $h_1, ..., h_l$, where h_i is a path from the planning graph $\mathbf{PG}_i$, representing history of the object O_i during execution the last meta action,
  for $i = 1, ..., l$,
- plan P_1 generated for a given structured object O before reconstruction,
- position *pos* of starting state of reconstruction in the plan P_1.

Output: The plan P_1 after reconstruction

```
1  Procedure $MReconstruction(H, \{h_1, ..., h_l\}, P_1, pos)$
2  begin
3      $j := pos$
4      while $(j \leq pos + 2 \cdot (D_r - 1))$ do
5          $P_2 := EEFSS(H, \{h_1, ..., h_l\}, P_1[j], L_{rp})$
6          if ($P_2$ is not empty) then
7              $P_3 :=$
               $Subpath(P_1, 1, pos - 1) + P_2 + Subpath(P_1, j + 1, length(P_1))$
8              return $P_3$
9          end
10         $j := j + 2$
11     end
12     return "empty plan"
13 end
```

on pairs of medical plans (P_1, P_2) (of the same length) with the sequences of actions $\mathcal{A}(P_1) = (A_1, ..., A_n)$ and $\mathcal{A}(P_2) = (B_1, ..., B_n)$, respectively as follows

$$Sim(P_1, P_2) = \frac{1}{n} \sum_{i=1}^{n} \frac{|A_i \cap B_i| + |M \setminus (A_i \cup B_i)|}{|M|}.$$

However, such an approach seems to be very abstract and ad hoc, because it does not take into account any deeper knowledge about the similarity of plans, e.g., domain knowledge. Whereas, the similarity relations for real-life problems are usually more complex objects, i.e., their construction from local similarities cannot be obtained by simple fusion functions. Hence, such similarity relations cannot be approximated with the satisfactory quality by employing the existing simple strategies. For this reason we treat this similarity measure, Sim, only as an example and do not take into account in our further research (and in our proposed method). Whereas, to support the process of similarity relation approximation, we propose to use domain knowledge represented by concept ontology expressed in natural language. The ontology consists of concepts used by expert in his explanation of similarity and dissimilarity cases. Approximation of the ontology makes it possible to obtain some relevant concepts for approximation of the similarity relation.

7.19 Ontology of the Similarity between Plans

According to the domain knowledge, it is quite common, that there are many aspects of similarity between plans. For example, in case of comparison of medical plans used for the treatment of infants with respiratory failure, we should take into consideration, *e.g.*, the similarity of the antibiotics use, the ventilation mode and the similarity of PDA closing (see Appendix B for mor medical details). Moreover, every aspect of the similarity should be understood in a different way. For example, in estimation of the similarity in the antibiotic treatment, it should be evaluated the kind of antibiotic, as well as the time of administration. Therefore, it is necessary to investigate and take into account all incompatibilities of the antibiotic use between corresponding pairs of nodes from both plans. Excessive doses are rather acceptable (based on expert knowledge), whilst the lack of medicine (if it is necessary) should be taken as a very serious mistake. In such situation, the difference in our assessment is estimated as very significant. A bit different interpretation of similarity should be used in case of the ventilation. As in antibiotic use, we investigate all incompatibilities of the ventilation mode between corresponding pairs of nodes from both plans. However, sometimes, according to expert knowledge, we simplified our assessments, *e.g.*, respiration unsupported and CPAP are estimated as similar (see Example 36 for more medical details). More complicated situation is present if we want to judge the similarity in treatment of PDA. We have to assign the ventilation mode, as well as the similarity of PDA closing procedure. In summary, any aspect of the similarity between plans should be taken into account in the specific way and the domain knowledge is necessary for joining all these similarities (obtained for all aspects). Therefore, the similarity between plans should be assigned on the basis of a special ontology specified in a dialog with human experts. Such ontology we call *similarity ontology.* Using such similarity ontology we developed methods for inducing classifiers predicting the similarity between two plans (generated automatically and proposed by human experts).

In the paper, we assume that each similarity ontology between plans has a tree structure. The root of this tree is always one concept representing general similarity between plans. In each similarity ontology there may exist concepts of two-way type. In this paper, the concepts of the first type will be called *internal concepts* of ontology. They are characterized by the fact that they depend on other ontology concepts. The concept of the second type will be called *input concepts* of ontology (in other words the concepts of the lowest ontology level). The input concepts are characterized by the fact that they do not depend on other ontology concepts. Fig. 50 shows an exemplary ontology of similarity between plans of the treatment of newborn infants with the respiratory failure.

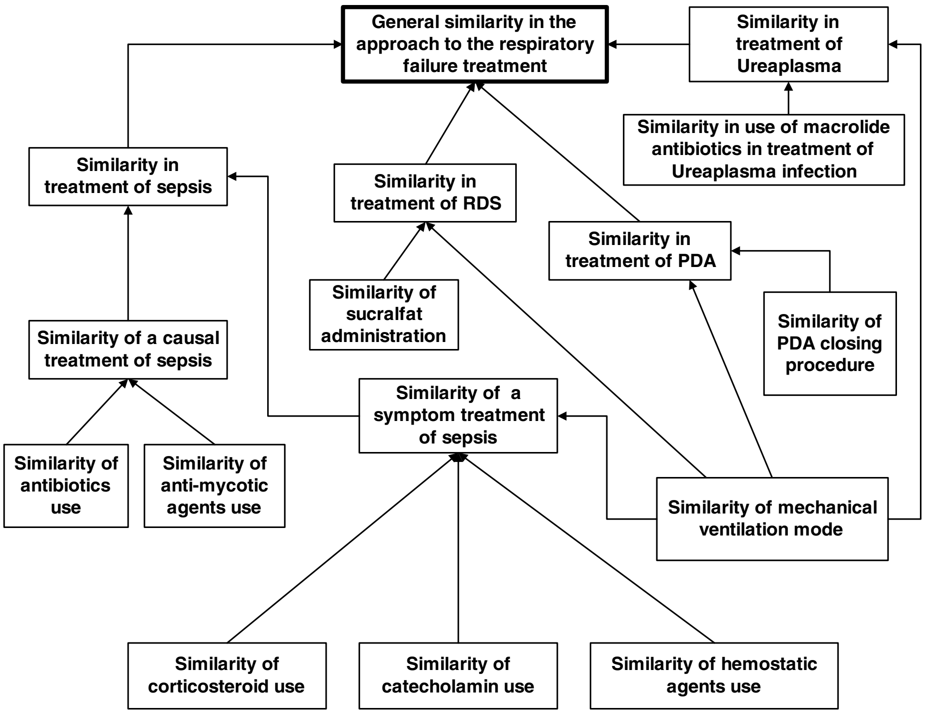

General similarity in the approach to the respiratory failure treatment
Similarity in treatment of Ureaplasma
Similarity in use of macrolide antibiotics in treatment of Ureaplasma infection
Similarity in treatment of sepsis
Similarity in treatment of RDS
Similarity in treatment of PDA
Similarity of sucralfat administration
Similarity of PDA closing procedure
Similarity of a causal treatment of sepsis
Similarity of a symptom treatment of sepsis
Similarity of antibiotics use
Similarity of anti-mycotic agents use
Similarity of mechanical ventilation mode
Similarity of corticosteroid use
Similarity of catecholamin use
Similarity of hemostatic agents use

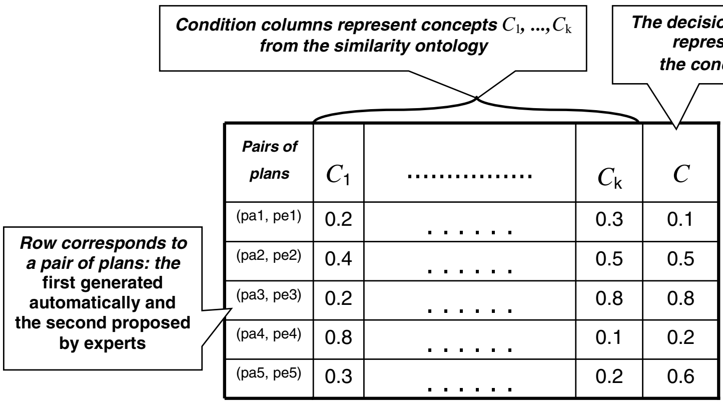
Fig. 50. An exemplary ontology of similarity between plans of the treatment of new-born infants with respiratory failure

This ontology has been provided by human experts. However, it is also possible to present some other versions of such ontology, instead of that presented above, according to opinions of some other group of human experts.

7.20 Similarity Classifier

Using the similarity ontology (*e.g.*, the ontology presented in Fig. 50), we developed methods for inducing classifiers predicting the similarity between two plans (generated automatically and proposed by human experts).

The method for construction of such classifier can be based on *a similarity table of plans*. The similarity table of plans is the decision table which may be constructed for any concept from the similarity ontology. The similarity table is created in order to approximate a concept for which the table has been constructed. The approximation of the concept takes place with the help of classifiers generated for the similarity table. However, because of the fact that in the similarity ontology there occur two types of concepts (internal and input), there are also two types of similarity tables. Similarity tables of the first type are constructed for internal concepts, whereas the tables of the second type are constructed for input concepts.


Condition columns represent concepts $C_1, ..., C_k$ from the similarity ontology
The decision column represents the concept C
Row corresponds to a pair of plans: the first generated automatically and the second proposed by experts

Pairs of plans	C_1	....	C_k	C
(pa1, pe1)	0.2	......	0.3	0.1
(pa2, pe2)	0.4	......	0.5	0.5
(pa3, pe3)	0.2	......	0.8	0.8
(pa4, pe4)	0.8	......	0.1	0.2
(pa5, pe5)	0.3	......	0.2	0.6

Fig. 51. The scheme of the similarity table of plans

Similarity tables for internal concepts of similarity ontology are constructed for a certain fragment of similarity ontology which consists of a concept of this ontology and concepts on which this concept depends. In the case of ontology from Fig. 50 it may be for instance the concept *Similarity of a symptom treatment of sepsis* and concepts *Similarity of corticosteroid use*, *Similarity of catecholamin use* and *Similarity of hemostatic agents use.* To simplify further discussion let us assume that it is the concept C that depends in the similarity ontology on the concepts C_1, ..., C_k. The aim of constructing a similarity table is approximation of concept C using concepts C_1, ..., C_k (see Fig. 51). Condition columns of such similarity table represent concepts C_1, ..., C_k. Any row corresponds to a pair of plans: generated automatically and proposed by experts. Values of all attributes have been provided by experts from the set $\{0.0, 0.1, ..., 0.9, 1.0\}$. Finally, the decision column represents the concept C.

Definition 71. *Let us assume that:*

- $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t}, a_c)$ *is a temporal information system with actions,*
- $\mathbf{PG} = (S, A, E)$ *is a planning graph,*
- k *is a fixed length of plans in the planning graph* $\mathbf{PG}$*,*
- C *is an internal concept, dependent in some similarity ontology on the concepts* C_1,...,C_k*, where* $C, C_1, ..., C_k \subseteq PLAN(\mathbf{PG}, k) \times PLAN(\mathbf{PG}, k)$.

A similarity table of plans from planning graph $\mathbf{PG}$ *constructed for the internal concept* C *on the basis of the system* $\mathbf{T}$ *is a decision table* (U, A, d)*, where:*

- $U \subseteq DPLAN(\mathbf{T}, \mathbf{PG}, k) \times DPLAN(\mathbf{T}, \mathbf{PG}, k)$,
- $A = \{a_1, ..., a_m\}$ *is the set of attributes created on the basis of concepts from the family* $C_1, ..., C_m$ *such that for any* $i \in \{1, ..., m\}$ *values of* a_i *describe membership of objects from the set* U *to the concept* C_i *and these values are determined by experts from the set* $\{$ 0.0, 0.1, ..., 1.0 $\}$,

– *the values of decision attribute d which also belong to the set* { 0.0, 0.1, ..., 1.0 } *are proposed on the basis of the dissimilarity function value (proposed by an expert) of plans for individual objects from the set U.*

Let us notice that in the similarity table defined above there are no all the possible pairs of plans from set $DPLAN(\mathbf{T}, \mathbf{PG}, k) \times DPLAN(\mathbf{T}, \mathbf{PG}, k)$, but only a certain selected subset of the set of these pairs. In practice, this limitation is very necessary because the number of pairs of product $DPLAN(\mathbf{T}, \mathbf{PG}, k)$ $\times$ $DPLAN(\mathbf{T}, \mathbf{PG}, k)$ may be so large that the expert is not able to provide for them all values of decision attribute d. Therefore, usually in the similarity table there are only pairs selected by the expert which represent typical cases of determining similarity functions of plans which may be generalized using a classifier.

The stratifying classifier computed for a similarity table (called *a similarity classifier*) can be used to determine the similarity between plans (generated by our methods of automated planning and plans proposed be human experts) relatively to a given internal concept C.

Such stratifying classifiers may be constructed for all concepts from the similarity ontology which depend, in this ontology, on other concepts. However, we also need stratifying classifiers for input concepts of ontology, that is, those lying on the lowest level of the ontology. Hence, they are the concepts which do not depend on other concepts in this ontology. To approximate them we do not use other ontology concepts but we apply the features of comparable plans which are expressed in the form of patterns defined in the language $FPPG$ (see Section 7.6). Obviously, such types of patterns are also concepts determined in the set of pairs of plans. However, they are usually not placed in the similarity ontology between plans. Therefore, approximation tables of input concepts of the similarity ontology should be treated as a specific type of similarity table.

Definition 72. *Let us assume that:*

– $\mathbf{T} = (U, A, a_{id}, \leq_{a_{id}}, a_t, \leq_{a_t}, a_c)$ *is a temporal information system with actions,*
– $\mathbf{PG} = (S, A, E)$ *is a planning graph,*
– k *is a fixed length of plans in the planning graph* $\mathbf{PG}$*,*
– $\phi_1, ..., \phi_m \in FPPG(\mathbf{PG})$ *is a family of formulas defined by experts,*
– $C \subseteq PLAN(\mathbf{PG}, k) \times PLAN(\mathbf{PG}, k)$ *is an input concept of the similarity ontology between plans.*

A similarity table of plans from planning graph $\mathbf{PG}$ *constructed for the input concept C on the basis of the system* $\mathbf{T}$ *is a decision table* (U, A, d)*, where:*

– $U \subseteq DPLAN(\mathbf{T}, \mathbf{PG}, k) \times DPLAN(\mathbf{T}, \mathbf{PG}, k)$,
– $A = \{a_1, ..., a_m, a_{m+1}, ..., a_{2m}\}$ *is a set of attributes created on the basis of formulas $\phi_1, ..., \phi_m$, where for any $i \in \{1, ..., 2m\}$ values of a_i are computed in the following way:*

$$\forall p = (p_1, p_2) \in U : a_i(p) = \begin{cases} 1 & \text{if } i \leq m \text{ and } p_1 \models_{FPPG(\mathbf{PG})} \phi_i \\ 1 & \text{if } i > m \text{ and } p_2 \models_{FPPG(\mathbf{PG})} \phi_i \\ 0 & \text{otherwise} \end{cases},$$

– *the values of decision attribute d describe membership of objects from the set U to the concept C and these values are determined by experts from the set* $\{ 0.0, 0.1, ..., 1.0 \}$.

Let us notice that the similarity table defined above is constructed in the way that concept C is approximated on the basis of the features of both plans corresponding to a given object from the set U.

It is worth noticing that for approximation of complex concepts from the similarity ontology one can use also features (attributes) describing relations between plans. Such features are formulated in a natural language using special questions about both plans. Examples of such questions are: *Were antibiotics used simultaneously in both plans?*, *Was the average difference between mechanical ventilation mode in both plans significant?*. However, it requires a simple extension of the language $FPPG(\mathbf{PG})$.

Classifiers constructed for similarity tables corresponding to all concepts from the similarity ontology may be used to construct a complex classifier which gives the general similarity between plans (represented by the concept lying in the root of the similarity ontology). We provide an example of how such a classifier works. Let us assume that there is a certain similarity ontology between pairs of plans in which there occur six following concepts: C_1, C_2, C_3, C_4, C_5 and C_6. The concept C_1 depends on concepts C_2 and C_3, the concept C_2 depends on concepts C_4 and C_5, and the concept C_3 depends on concepts C_5 and C_6. In this ontology concept C_1 is the concept of general similarity between plans, whereas concepts C_4, C_5 and C_6 are input concepts of the similarity ontology (see Fig. 52).

Firstly, we construct similarity tables for concepts C_4, C_5, C_6 and stratifying classifiers μ_{C_4}, μ_{C_5}, μ_{C_6} corresponding to them. Let us also assume that there are given stratifying classifiers μ_{C_1}, μ_{C_2}, μ_{C_3} which were constructed for similarity tables which correspond to concepts C_1, C_2 and C_3. Tested object $u = (p_1, p_2)$ which is a pair of compared plans is classified to the layer of concept C corresponding to it in the following way. At the beginning, the object u is classified by classifiers μ_{C_4}, μ_{C_5} and μ_{C_6}. This way we obtain values $\mu_{C_4}(u)$, $\mu_{C_5}(u)$ and $\mu_{C_6}(u)$. Next, values $\mu_{C_4}(u)$ and $\mu_{C_5}(u)$ are used as the values of conditional attributes in the similarity table constructed for concept C_2. Thus, the object u may be classified by classifier μ_{C_2}, which gives us value $\mu_{C_2}(u)$. At the same

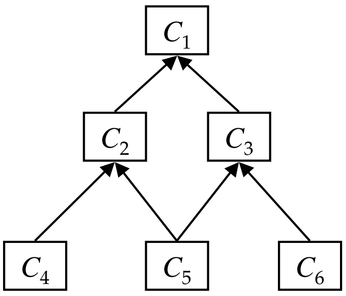


Fig. 52. The scheme of a simple similarity ontology

time, values $\mu_{C_5}(u)$ and $\mu_{C_6}(u)$ are used as the values of conditional attributes in the similarity table constructed for concept C_3. It gives the possibility to classify object u by classifier μ_{C_3} and obtain value $\mu_{C_3}(u)$. Finally, values $\mu_{C_2}(u)$ and $\mu_{C_3}(u)$ are used as the values of conditional attributes of the similarity table constructed for concept C_1. Thus, the object u may be classified by classifier μ_{C_1} to layer $\mu_{C_1}(u)$.

The complex classifier described above can be used to determine the general similarity between plans generated by our methods of automated planning and plans proposed by human experts, *e.g.*, during the real-life clinical treatment (see Section 7.21).

7.21 Experiments with Medical Data

To verify the effectiveness of presented in this paper methods of automated planning, we have implemented the algorithms in an *Automated Planning* library (AP-lib), which is an extension of the RSES-lib library forming the computational kernel of the RSES system (see [15]).

It should be emphasized that, in general, automated planning of treatment is a very difficult and complicated task because it requires extensive medical knowledge combined with sensor information about the state of a patient. Even so, the proposed approach makes it possible to obtain quite satisfactory results in the short-term planning of treatment of infants with respiratory failure. The reason is that medical data sets have been accurately prepared for purposes of our experiments using the medical knowledge. For example, the collection of medical actions, that are usually used during the treatment of infants with respiratory failure, has been divided into a few groups of similar actions (for example: antibiotics, anti-mycotic agents, mechanical ventilation, catecholamines, corticosteroids, hemostatic agents). It is very helpful in the prediction of actions because the number of actions is significantly decreased.

The experiments have been performed on the medical data sets obtained from Neonatal Intensive Care Unit, First Department of Pediatrics, Polish-American Institute of Pediatrics, Collegium Medicum, Jagiellonian University, Krakow, Poland (see also Section 6.26). We used one data table, that consists of 11099 objects. Each object of this table describes parameters of one patient in single time point. There were prepared 7022 situations on the basis of this data table, where the plan of treatment has been proposed by human experts during the real-life clinical treatment.

We have applied the *train-and-test* method. However, analogously to experiments from Subsection 6.26 the method of dividing data differed slightly from the standard method described in Section 2.9. Namely, in each experiment the whole set of patients was randomly divided into two groups (training and tested one). Each of these groups allowed creating approximately 4000 time windows which have duration of 7 time points. Time windows created on the basis of patients from the training part created a training table for a given experiment (when plans of treatment have been assigned), whereas time windows created on the basis of patients from the tested part created a test table for the experiment (when plans have been generated by automated method and expert plans are known in order to compare both plans).

In the discussed experiments, the distance between time points recorded for a specific patient was constant (one day). In a single experiment concerning a patient's treatment, a 7-point sequence of time points was used. In terms of planning the treatment each such sequence may be written as s_1, a_1, s_2, a_2, s_3, a_3, s_4, a_4, s_5, a_5, s_5, a_6, s_7, where s_i (for $i = 1, ..., 7$) is a patient state and a_i (for $i = 1, ..., 6$) is a complex medical action performed in the state s_i. The first part of the above sequence of states and actions, that is, from state s_1 to state s_3, was used by the method of automated planning as the input information (corresponding to the values of conditional attributes in the classic approach to constructing classifiers). The remaining actions and states were automatically generated to create plan (s_3, a'_3, s'_4, a'_4, s'_5, a'_5, s'_6, a'_6, s'_7). This plan may be treated as a certain type of a complex decision value. Verification of the quality of the generated plan consisted in comparing plan (s_3, a'_3, s'_4, a'_4, s'_5, a'_5, s'_6, a'_6, s'_7) with plan (s_3, a_3, s_4, a_4, s_5, a_5, s_5, a_6, s_7). It is worth adding that a single complex action concerned one time point, meta action concerned two time points and a single experiment consisted in planning two meta actions. Hence, in a single experiment four actions were planned (patient's treatment for four days). In other words, at the beginning of the automated planning procedure the information about the patient's state in the last three days of his hospitalization was used (s_1, s_2, s_3) together with the information about complex medical actions undertaken one or two days before (a_1, a_2). The generated plan included information about a suggested complex medical action on a given day of hospitalization (a'_3), information about actions which should be undertaken in the three following days of hospitalization (a'_4, a'_5, a'_6) and information about the patient's state anticipated as a result of the planned treatment in the four following days of hospitalization (s'_4, s'_5, s'_6, s'_7).

As a measure of planning success (or failure) in our experiments, we use the special classifier that can predict the similarity between two plans as a number between 0.0 (very low similarity between two plans) and 1.0 (very high similarity between two plans) (see Section 7.20). We use this classifier to determine the similarity between plans generated by our methods of automated planning and plans proposed be human experts during the real-life clinical treatment. In order to determine the standard deviation of the obtained results each experiment was repeated for 10 random divisions of the whole data set.

The average similarity between plans for all tested situations was **0.802**. The corresponding standard deviations was **0.041**. The coverage of tested situation by generated plans was **0.846** with standard deviation **0.018**.

Due to the fact that the average similarity is not too high (less than 0.9) and the standard deviation is relatively high for our algorithm, we present also the distribution of the results. We describe results in such a way that we present how many generated plans belong to the specified interval of similarity. For this reason we divided interval [0.0, 1.0] into 5 equal intervals, i.e., [0.0, 0.2], [0.2, 0.4], [0.4, 0.6], [0.6, 0.8] and [0.8, 1.0]. Table 10 shows the average percent of the

Table 10. The average percent of plans belonging to the specified interval and the average similarity of plans in this interval

Intervals	Average percent of plans	Average similarity of plans
[0.0, 0.2]	**12.1%** ± 4.5%	**0.139** ± 0.002
(0.2, 0.4]	**6.2%** ± 1.5%	**0.349** ± 0.003
(0.4, 0.6]	**7.1%** ± 1.7%	**0.563** ± 0.002
(0.6, 0.8]	5.8% ± 0.9%	0.773 ± 0.004
(0.8, 1.0]	68.9% ± 5.6%	0.987 ± 0.002

plans belonging to the specified interval and the average similarity of plans in this interval.

It is easy to see that some group of plans generated automatically is not enough similar to the plans proposed by the experts. If we assume that inadequate similarity is lower than 0.6, in this group we found about 25% of all plans (see Table 10). To explain this issue, we should observe more carefully plans, which are incompatible with the proposals prepared by experts. In practice, the main medical actions influencing the similarity of plans in accordance with ontology of the similarity from Fig. 50 are mechanical ventilation, antibiotics, anti-mycotic agents and macrolide antibiotics. Therefore, it may be interesting how the treatment similarity changed in the range of applying these actions in the individual intervals of similarity between the plans.

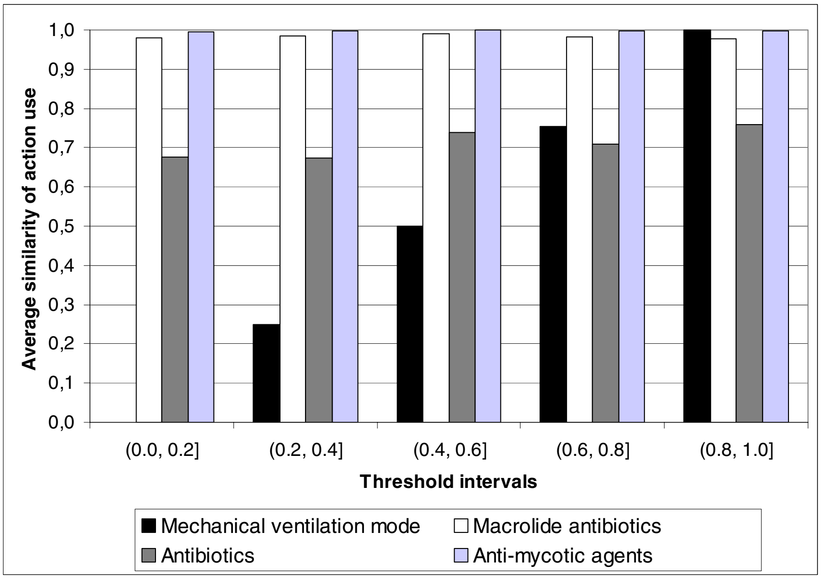


Fig. 53. The average similarity of plans in the specified interval for medical actions

On Fig. 53 we can see that a significant incompatibility of treatment plans most often concerns mechanical ventilation and perhaps antibiotic therapy - the situation when a patient develops a sudden and severe infection (*e.g.*, sepsis). Such circumstances cause rapid exacerbation of respiratory failure are required higher level of mechanical ventilation and immediate antibiotic treatment. For example, although microbiological confirmation of current infection is achieved after 2–3 days, physician starts treatment after first symptoms of suspected disease and often intensify mechanical ventilation mode. It would seem that the algorithms of automated planning presented in this paper may imitate the strategy of treatment described above. Unfortunately, in practice, these algorithms are not able to learn this strategy for a lot of information because they were not introduced to the base records or were introduced with delay. For instance, hemoglobin saturation which is measured for the whole time, as the dynamic marker of patients respiratory status, was not found in the data, whilst results of arterial blood gases were introduced irregularly, with many missing values. So, the technical limitation of the current data collection lead to the intensive work modifying and extending both, the equipment and software, served for gathering clinical data. It may be expected that in several years the automated planning algorithms, described in this paper, will achieve much better and useful results.

A separate problem is a relatively low coverage of the algorithms described in this paper which equals averagely 0.846. Such a low coverage results from the specificity of the automated planning method used which synchronizes the treatment of four diseases (RDS, PDA, sepsis and Ureaplasma). We may identify two reasons of a low coverage. Firstly, because of data shortage the algorithm in many situations may not synchronize the treatment of the above mentioned diseases. It happens this way because each proposed comparison of plans may be debatable in terms of the knowledge gathered in the system. Therefore, in these cases the system does not suggest any treatment plan and says *I do not know.* The second reason for low coverage is the fact that the automated planning method used requires application of a complex classifier which consists of many classifiers of lesser complexity. Putting these classifiers together often causes the effect of decreasing the complex classifier coverage. For instance, let us assume that making decision for tested object u requires application of complex classifier μ, which consists of two classifiers μ_1 and μ_2. We apply classifier μ_1 directly to u, whereas classifier μ_2 is applied to the results of classification of classifier μ_1. In other words, to make classifier μ_2 work for a given tested object u we need value $\mu_1(u)$. Let us assume that the coverage for classifiers μ_1 and μ_2 equals respectively 0.94 and 0.95. Hence, the coverage of classifier μ is equal $0.94 \cdot 0.95 = 0.893$, that is the coverage of classifier μ is smaller than the coverage of classifier μ_1 as well as the coverage of classifier μ_2.

It is worth noticing that applying other automated planning methods (see [353, 354]), a higher coverage for the data sets analyzed here may be obtained. However, in this paper we prefer algorithms *EEFS* and *EEFSS*, because they need not only data sets but also great domain knowledge to work. Moreover, the quality of their performance depends greatly on the provided domain knowledge. Therefore, it may be expected that providing greater and more reliable knowledge and more extensive data sets on the input of this algorithm will allow the quality of automatically generated treatment plans to be more similar to the quality of treatment plans proposed by the medical experts.

In summation, we conclude that experimental results showed that the proposed automated planning method gives good results, also in the opinion of medical experts (compatible enough with the plans suggested by the experts), and may be applied in medical practice as a supporting tool for planning the treatment of infants suffering from respiratory failure.

8 Summary

The aim of this paper was to present new methods of approximating complex concepts on the basis of experimental data and domain knowledge which is mainly represented using concept ontology.

At the beginning of the paper a number of methods of constructing classical classifiers were overviewed (see Section 2) and methods of constructing stratifying classifiers were presented (see Section 3). Next, in Section 4, a general methodology of approximating complex concepts with the use of data sets and domain knowledge was presented. In the further part of the paper, this methodology was applied to approximate spatial complex concepts (see Section 5), spatio-temporal complex concepts for unstructured and structured objects (see Section 6), to identify the behavioral patterns for this type of objects (see Section 6), and to the automated planning of behavior of such objects when the states of objects are represented by spatio-temporal concepts which require an approximation (see Section 7).

We have also described the results of computer experiments conducted on real-life data sets which were obtained from the road traffic simulator (see Appendix A) and on medical data which were made available by Neonatal Intensive Care Unit, First Department of Pediatrics, Polish-American Institute of Pediatrics, Collegium Medicum, Jagiellonian University, Krakow, Poland.

In light of theoretical discourse and the results of computer experiments presented in the paper the following conclusions may be drawn:

1. The methodology of approximating complex concepts with the use of data sets and domain knowledge (represented mainly by a concept ontology), which was proposed in this paper (see Section 4), fills the gap which exists between spatio-temporal complex concepts and sensor data. It enables an effective approximation of complex concepts; however, it requires a domain knowledge represented mainly in the form of a concept ontology.
2. Stratifying classifiers proposed in the paper (see Section 3) are effective tools for stratifying concepts, that is, they enable to classify objects to different layers of the concept corresponding to different degrees of certainty of membership of a tested object to the concept. Particularly noteworthy here is the new method of constructing such classifiers based on shortening of

decision rules with respect to different coefficients of consistency. It makes an automatic stratification of concepts possible.

3. The method of approximation of complex spatial concepts, described in the paper, with the help of approximate reasoning schemes (AR-schemes) leads to better results than the classical methods based on decision rules induced directly from sensor data because the quality of classifier classification based on AR-schemes is higher than the quality of classification obtained by classifiers based on decision rules, particularly for small decision classes representing atypical cases in the recognition of which we are most interested in, *e.g.*, a dangerous driving vehicle on a highway (see Section 5). Moreover, for larger data sets, the time of constructing classifiers based on AR-schemes is much shorter than the time of inducing classifiers based on decision rules, and the structure of classifiers based on AR-schemes is less complex than the structure of classifiers based on decision rules. It is also worth mentioning that the classifiers based on AR-schemes are more robust (stable or tolerant) when it comes to changes in training data sets serving the construction of classifiers, that is, a classifier based on AR-schemes, constructed for one data set, often proves itself good for another data set. For example, a classifier constructed for data generated from the traffic simulator with one simulation scenario proves itself useful in classification of objects generated by the simulator with the use of another simulation scenario.
4. The methodology of modeling complex object behavior with the use of behavioral graphs of these objects, proposed in the paper (see Section 6), is a convenient and effective tool for identifying behavioral patterns of complex objects. On the one hand this methodology, enables to represent concepts on a high abstraction level, and on the other hand, owing to the use of a domain knowledge, it enables to approximate these concepts on the basis of sensor data and using a domain knowledge.
5. The sweeping method around complex objects is a very fast and convenient method of isolating objects with complex structure from complex dynamical systems (see Section 6). A certain difficulty in using this method is the fact that each of its applications requires appropriate sweeping heuristics which must be proposed by the expert.
6. The method of eliminating complex objects in behavioral pattern identification based on the rules of fast elimination of behavioral patterns greatly accelerates the identification of behavioral patterns in complex dynamical systems monitoring (see Section 6). A certain inconvenience of applying this method is the fact that although it is based on rules of fast elimination of behavioral patterns obtained from data, in order to determine those rules we need special temporal patterns supporting the process of elimination, and the patterns themselves must be proposed by experts.
7. The methods of automated planning of complex object behavior proposed in the paper facilitate an effective planning of behavior of both unstructured and structured objects whose states are defined in a natural language using vague spatio-temporal conditions (see Section 7). The authenticity of conditions of this type is usually not possible to be verified on the basis

of a simple analysis of available information about the object and that is why these conditions must be treated as spatio-temporal complex concepts and their approximation requires methods described in this paper which are based on data sets and domain knowledge.

8. The method of solving conflicts between actions during automated planning of complex object behavior, based on a domain knowledge and data sets (see Section 7), is ideal for the situation where the choice of one action from many possible actions, to be performed at a given state should not be random but it should be made in accordance with the domain knowledge (as in the example of planning of a patient's treatment). A significant novelty of the method presented here in relation to the already existing ones is the fact that in order to solve conflicts between actions we use classifiers based on data sets and domain knowledge.
9. The method of synchronizing plans constructed for parts of a structured object, described in the paper, is an important element of the method of planning of structured object behavior (see Section 7). If we assume that plans constructed for parts of a structured object are processes of some kind, then the method of synchronizing those plans is a method of synchronizing processes corresponding to the parts of the structured object. It should be emphasized, however, that the significant novelty of the method of synchronizing processes presented herein in relation to the ones known from literature is the fact that the synchronization is carried out by using classifiers determined on the basis of data sets and domain knowledge.
10. The method of partial reconstruction of a plan, proposed in the paper, can accelerate the execution of plans constructed for complex objects significantly (see Section 7). It is due to the fact that in the case of incompatibility of the state occurring in the plan with the actual state of the complex object, the plan needs not be generated from the beginning but it may be reconstructed using the repair plan.
11. The method of similarity relation approximation based on data set usage and a domain knowledge expressed in the form of a concept ontology which has frequently been used throughout our research is an important proposal of solving a difficult problem of similarity relation approximation.

The main advantage of the methods of construction of classifiers for spatial and spatio-temporal complex concepts, presented in the paper, is the fact that besides data sets they also intensely use a domain knowledge expressed, above all, in the form of a concept ontology. Although using a domain knowledge causes an increase in effectiveness of the methods presented in this paper, it can be, however, a source of many difficulties arising during the application of those methods. The difficulties arise from the fact that the domain knowledge is often available only to the experts from the given domain. Therefore, projects which use the methods described in this paper require an active participation of experts from different domains. Hence, difficulties may often occur to incline experts who are consent to devote their time to participate in a computer science project.

In order to overcome these difficulties, there is a need of construction of special methods of data analysis which might be useful in automatic search of knowledge to replace, at least partially, the domain knowledge obtained from experts. In the case of methods presented herein, we mean the methods of data analysis which allow an automatic detection of concepts, significant for construction of complex classifiers. For example, we are concerned with the following data analysis methods here:

1. Automatic detection of spatial or spatio-temporal concepts which may be used in approximating other more complex concepts.
2. Automatic detection of complex spatio-temporal concepts occurring in behavioral patterns (understood as behavioral graphs) and temporal dependencies among those concepts.
3. Automatic detection of complex spatio-temporal concepts which are states of unstructured or structured objects in automated planning of behavior of such objects.
4. Automatic detection of complex spatio-temporal concepts which are meta-actions to be performed for structured objects in automated planning of behavior of such objects.

We plan to construct the data analysis methods mentioned above in the future.

In summation, it may be concluded that in executing real-life projects related to the construction of the intelligent systems supporting decision-making, apart from data sets it is necessary to apply domain knowledge. Without its application successful execution of many such projects becomes extremely difficult or impossible. On the other hand, appropriate space must be found for the automated methods of classifier construction wherever it is feasible. It means, thus, finding a certain type of "the golden mean" to apply appropriate proportions in domain knowledge usage and automated methods of data analysis. Certainly, it will determine the success or failure of many projects.

Acknowledgement

I would like to express my deepest gratitude to my mentor, Professor Andrzej Skowron for the inspiration and comprehensive support during writing this paper. It goes without saying that without his great involvement in scientific research and creative influence he exerts on the scientific development of the Group of Mathematical Logic at the Institute of Mathematics, Warsaw University which remains under his supervision, this work would never have been written.

I am grateful to my wife, Anna, for her love and unconditional support which, without a doubt, were some of the main factors without which this paper would not be possible.

I would like to extend my gratefulness to Professor James F. Peters (Department of Electrical and Computer Engineering, University of Manitoba, Canada), Dr. Anna Gomolińska (Department of Mathematics, University of Białystok, Poland) and Dr. Piotr Synak (Polish-Japanese Institute of Information Technology, Warsaw, Poland) for their insights and many helpful comments and corrections made in this paper.

I would also like to thank Professor Jacek Pietrzyk and Dr. Piotr Kruczek (both from First Department of Pediatrics, Polish-American Institute of Pediatrics, Collegium Medicum, Jagiellonian University, Krakow, Poland) for providing for computer experiments valuable medical data which have been collected for many years in Neonatal Intensive Care Unit, First Department of Pediatrics, Polish-American Institute of Pediatrics, Collegium Medicum, Jagiellonian University, Krakow, Poland.

Special gratitude also goes to my sister Dr. Stanisława Bazan-Socha (Second Department of Internal Medicine, Collegium Medicum, Jagiellonian University, Krakow, Poland) and Dr. Piotr Kruczek for many invaluable discussions which helped me enormously to understand medical issues which was indispensable in preparation of medical data for the need of computer experiments.

Many of the results presented or mentioned in this paper are the results of long cooperation with other co-authors including Stanisława Bazan-Socha, Grzegorz Góra, Piotr Kruczek, Rafał Latkowski, Hung Son Nguyen, Sinh Hoa Nguyen, Antoni Osmólski, James F. Peters, Jacek J. Pietrzyk, Andrzej Skowron, Jarosław Stepaniuk, Piotr Synak, Marcin S. Szczuka, Dominik Ślęzak, Roman Świniarski, Hui Wang, Arkadiusz Wojna, Marcin Wojnarski, Jakub Wróblewski. At this point I want to thank them all for effective and pleasurable cooperation.

The research has been partially supported by the grant 3 T11C 002 26 from Ministry of Scientific Research and Information Technology of the Republic of Poland, by the grant N N516 368334 from Ministry of Science and Higher Education of the Republic of Poland and by the grant Innovative Economy Operational Programme 2007-2013 (Priority Axis 1. Research and development of new technologies) managed by Ministry of Regional Development of the Republic of Poland.

References

1. Murray, J.A., Bradley, H., Craigie, W., Onions, C.: The Oxford English Dictionary. Oxford University Press, Oxford (1933)
2. Devlin, K.: Logic and Information. Cambridge University Press, Cambridge (1991)
3. Joseph, H.W.B.: An Introduction to Logic. Clarendon Press, Oxford (1916)
4. Ogden, C.K., Richards, I.A.: The Meaning of Meaning. A Study of the Influence of Language Upon Thought and of the Science of Symbolism. Harcourt, Brace and Company, New York (1923)
5. Mendelson, E.: Introduction to Mathematical Logic. International Thomson Publishing (1987)
6. Friedman, J., Hastie, T., Tibshirani, R.: The Elements of Statistical Learning: Data Mining, Inference and Prediction, vol. I-V. Springer, Heidelberg (2001)
7. Gabbay, D.M., Hogger, C.J., Robinson, J.A. (eds.): Handbook of Logic in Artificial Intelligence and Logic Programming, vol. I-V. Oxford University Press, New York (1994)

8. Ignizio, J.P.: An Introduction to Expert Systems. McGraw-Hill, New York (1991)
9. Kloesgen, E., Zytkow, J. (eds.): Handbook of Knowledge Discovery and Data Mining. Oxford University Press, Oxford (2002)
10. Michalski, R., et al. (eds.): Machine Learning, vol. I-IV. Morgan Kaufmann, Los Altos (1983, 1986, 1990, 1994)
11. Michie, D., Spiegelhalter, D.J., Taylor, C.C.: Machine learning, neural and statistical classification. Ellis Horwood Limited, England (1994)
12. Mitchel, T.M.: Machine Learning. McGraw-Hill, Boston (1997)
13. Ripley, B.D.: Pattern Recognition and Neural Networks. Cambridge University Press, Cambridge (1996)
14. Bazan, J.G., Szczuka, M.: The Rough Set Exploration System. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets III. LNCS, vol. 3400, pp. 37–56. Springer, Heidelberg (2005)
15. RSES: Project web site, `http://logic.mimuw.edu.pl/~rses`
16. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning about Data. D: System Theory, Knowledge Engineering and Problem Solving, vol. 9. Kluwer Academic Publishers, Dordrecht (1991)
17. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Information Sciences 177, 3–27 (2007)
18. Peters, J.F., Skowron, A.: Zdzisław Pawlak life and work (1926–2006). Information Sciences 177, 1–2 (2007)
19. Brown, E.M.: Boolean Reasoning. Kluwer Academic Publishers, Dordrecht (1990)
20. Pawlak, Z., Skowron, A.: Rough sets and boolean reasoning. Information Sciences 177, 41–73 (2007)
21. Skowron, A., Rauszer, C.: The discernibility matrices and functions in information systems. In: Słowiński, R. (ed.) Intelligent Decision Support - Handbook of Applications and Advances of the Rough Sets Theory. D: System Theory, Knowledge Engineering and Problem Solving, vol. 11, pp. 331–362. Kluwer Academic Publishers, Dordrecht (1992)
22. Borrett, S.R., Bridewell, W., Langley, P., Arrigo, K.R.: A method for representing and developing process models. Ecological Complexity 4, 1–12 (2007)
23. Bridewell, W., Langley, P., Todorovski, L., Dzeroski, S.: Inductive process modeling. Machine Learning (to appear, 2008)
24. Langley, P.: Cognitive architectures and general intelligent systems. AI Magazine 27, 33–44 (2006)
25. Langley, P., Cummings, K., Shapiro, D.: Hierarchical skills and cognitive architectures. In: Proceedings of the Twenty-Sixth Annual Conference of the Cognitive Science Society, Chicago, IL, pp. 779–784 (2004)
26. Langley, P., Laird, J.E.: Cognitive architectures: Research issues and challenges. Technical report, Institute for the Study of Learning and Expertise, Palo Alto, CA (2002)
27. Langley, P., Laird, J.E., Rogers, S.: Cognitive architectures: Research issues and challenges. Technical report, Computational Learning Laboratory, CSLI, Stanford University, Palo Alto, CA (2006)
28. Langley, P., Shiran, O., Shrager, J., Todorovski, L., Pohorille, A.: Constructing explanatory process models from biological data and knowledge. AI in Medicine 37, 191–201 (2006)
29. Pal, S.K., Polkowski, L., Skowron, A. (eds.): Rough-Neural Computing: Techniques for Computing with Words. Cognitive Technologies. Springer, Heidelberg (2003)

30. Pancerz, K., Suraj, Z.: Discovering concurrent models from data tables with the ROSECON system. Fundamenta Informaticae 60, 251–268 (2004)
31. Pancerz, K., Suraj, Z.: Discovery of asynchronous concurrent models from experimental tables. Fundamenta Informaticae 61(2), 97–116 (2004)
32. Pancerz, K., Suraj, Z.: Rough sets for discovering concurrent system models from data tables. In: Hassanien, A.E., Suraj, Z., Ślęzak, D., Lingras, P. (eds.) Rough Computing: Theories, Technologies and Applications. Idea Group, Inc. (2007)
33. Pat, L., George, D., Bay, S., Saito, K.: Robust induction of process models from time-series data. In: Fawcett, T., Mishra, N. (eds.) Proceedings of the Twentieth International Conference on Machine Learning, Washington, D.C, pp. 432–439. AAAI Press, Menlo Park (2003)
34. Skowron, A., Suraj, Z.: Discovery of concurrent data models from experimental tables: A rough set approach. In: Fayyad, U.M., Uthurusamy, R. (eds.) Proceedings of the First International Conference on Knowledge Discovery and Databases Mining (KDD 1995), pp. 288–293. AAAI Press, Menlo Park (1995)
35. Soar: Project web site, `http://sitemaker.umich.edu/soar/home`
36. Suraj, Z.: Discovery of concurrent data models from experimental tables. Fundamenta Informaticae 28, 353–376 (1996)
37. Suraj, Z.: The synthesis problem of concurrent systems specified by dynamic information systems. In: Polkowski, L., Skowron, A. (eds.) Rough Sets in Knowledge Discovery 2. Applications, Case Studies and Software Systems. Studies in Fuzziness and Soft Computing, pp. 418–448. Physica-Verlag, Heidelberg (1998)
38. Suraj, Z.: Rough set methods for the synthesis and analysis of concurrent processes. In: Polkowski, L., Tsumoto, S., Lin, T.Y. (eds.) Rough set methods and applications: new developments in knowledge discovery in information systems. Studies in Fuzziness and Soft Computing, pp. 379–488. Physica-Verlag, Heidelberg (2000)
39. Suraj, Z.: Discovering concurrent data models and decision algorithms from data: A rough set approach. International Journal on Artificial Intelligence and Machine Learning IRSI, 51–56 (2004)
40. Unnikrishnan, K.P., Ramakrishnan, N., Sastry, P.S., Uthurusamy, R.: Service-oriented science: Scaling escience impact. In: Proceedings of the Fourth KDD Workshop on Temporal Data Mining: Network Reconstruction from Dynamic Data, The Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data (KDD 2006), Philadelphia, USA, August 20-23 (2006)
41. Breiman, L.: Statistical modeling: the two cultures. Statistical Science 16(3), 199–231 (2001)
42. Poggio, T., Smale, S.: The mathematics of learning: Dealing with data. Notices of the American Mathematical Society (AMS) 5, 537–544 (2003)
43. Vapnik, V. (ed.): Statistical Learning Theory. Wiley, New York (1998)
44. Zadeh, L.A.: From computing with numbers to computing with words – from manipulation of measurements to manipulation of perceptions. IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications 1, 105–119 (1999)
45. Zadeh, L.A.: A new direction in AI: Toward a computational theory of perceptions. AI Magazine 1, 73–84 (2004)
46. Zadeh, L.A.: Toward a generalized theory of uncertainty (GTU) - an outline. Information Sciences 171, 1–40 (2005)

47. Ambroszkiewicz, S., Bartyna, W., Faderewski, M., Terlikowski, G.: An architecture of multirobot system based on software agents and the SOA paradigm. In: Czaja, L. (ed.) Proceedings of the Workshop on Concurrency, Specification, and Programming (CS&P 2007), Łagów, Poland, Warsaw, Poland, Warsaw University, September 27–29, pp. 21–32 (2007)
48. Domingos, P.: Toward knowledge-rich data mining. Data Mining and Knowledge Discovery 1, 21–28 (2007)
49. Foster, I.T.: Service-oriented science: Scaling escience impact. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Hong Kong, China, December 18-22. IEEE Computer Society, Los Alamitos (2006)
50. Kriegel, H.P., Borgwardt, K.M., Kröger, P., Pryakhin, A., Schubert, M., Zimek, A.: Future trends in data mining. Data Mining and Knowledge Discovery 1, 87–97 (2007)
51. Kuipers, B.: The spatial semantic hierarchy. Artificial Intelligence 119, 191–233 (2000)
52. Stone, P., Sridharan, M., Stronger, D., Kuhlmann, G., Kohl, N., Fidelman, P., Jong, N.K.: From pixels to multi-robot decision-making: A study in uncertainty. Robotics and Autonomous Systems 54, 933–943 (2006)
53. Guarino, N.: Formal ontology and information systems. In: Proceedings of the First International Conference on Formal Ontology in Information Systems (FOIS 1998), Trento, Italy, June 6-8, pp. 3–15. IOS Press, Amsterdam (1998)
54. Jarrar, M.: Towards Methodological Principles for Ontology Engineering. Ph.D thesis, Vrije Universiteit Brussel (2005)
55. Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A. (eds.): Feature Extraction Foundations and Applications. Studies in Fuzziness and Soft Computing, vol. 207. Springer-Verlag, Berlin (2006)
56. Liu, H., Motoda, H. (eds.): Feature Extraction, Construction and Selection: A Data Mining Perspective. The Springer International Series in Engineering and Computer Science, vol. 453. Springer, Berlin (1998)
57. Liu, H., Motoda, H. (eds.): Feature Selection for Knowledge Discovery and Data Mining. The Springer International Series in Engineering and Computer Science, vol. 454. Springer, Berlin (1998)
58. Dejong, G., Mooney, R.: Explanation-based learning: An alternative view. Machine Learning 1, 145–176 (1986)
59. Ellman, T.: Explanation-based learning: a survey of programs and perspectives. ACM Computing Surveys 21, 163–221 (1989)
60. Mitchell, T.M., Keller, R.M., Kedar-Cabelli, S.T.: Explanation-based generalization: A unifying view. Machine Learning 1, 47–80 (1986)
61. Mitchell, T.M., Thrun, S.B.: Learning analytically and inductively. In: Steier, D.M., Mitchell, T.M. (eds.) Mind Matters: A Tribute to Allen Newell, pp. 85–110. Lawrence Erlbaum Associates, Inc., Mahwah (1996)
62. Penczek, W., Pólrola, A.: Advances in Verification of Time Petri Nets and Timed Automata: A Temporal Logic Approach. Studies in Computational Intelligence. Springer, Secaucus (2006)
63. Roddick, J.F., Hornsby, K., Spiliopoulou, M.: An updated bibliography of temporal, spatial and spatio-temporal data mining research. In: Roddick, J.F., Hornsby, K. (eds.) TSDM 2000. LNCS, vol. 2007, pp. 147–163. Springer, Heidelberg (2001)

64. Roddick, J.F., Hornsby, K., Spiliopoulou, M.: Yabtsstdmr - yet another bibliography of temporal, spatial and spatio-temporal data mining research. In: Unnikrishnan, K.P., Uthurusamy, R. (eds.) SIGKDD Temporal Data Mining Workshop. ACM Press, pp. 167–175. Springer, San Francisco (2001)
65. Ichise, R., Shapiro, D., Langley, P.: Learning hierarchical skills from observation. In: Lange, S., Satoh, K., Smith, C.H. (eds.) DS 2002. LNCS, vol. 2534, pp. 247–258. Springer, Heidelberg (2002)
66. Makar, R., Mahadevan, S., Ghavamzadeh, M.: Hierarchical multi-agent reinforcement learning. In: Müller, J.P., Andre, E., Sen, S., Frasson, C. (eds.) Proceedings of the Fifth International Conference on Autonomous Agents, Montreal, Canada, pp. 246–253. ACM Press, New York (2001)
67. Paine, R.W., Tani, J.: Motor primitive and sequence self-organization in a hierarchical recurrent neural network. Neural Networks 17, 1291–1309 (2004)
68. Zhang, L., Zhang, B.: Hierarchical machine learning - a learning methodology inspired by human intelligence. In: Wang, G.-Y., Peters, J.F., Skowron, A., Yao, Y. (eds.) RSKT 2006. LNCS, vol. 4062, pp. 28–30. Springer, Heidelberg (2006)
69. Paine, R.W., Tani, J.: How hierarchical control self-organizes in artificial adaptive systems. Adaptive Behavior 13, 211–225 (2005)
70. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory and Practice. Elsevier, Morgan Kaufmann, CA (2004)
71. Haslum, P.: Modern AI planning: Reading list, `http://www.ida.liu.se/~pahas/maip/reading.ps`
72. ICAPS 2006: Proceedings of the Sixteenth International Conference on Automated Planning & Scheduling, The English Lake District, Cumbria, UK, June 6-10. AAAI Press, Menlo Park (2006)
73. LaValle, S.M.: Planning Algorithms. Cambridge University Press, New York (2006)
74. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice Hall, New Jersey (2003)
75. Vlahavas, I., Vrakas, D. (eds.): Intelligent Techniques for Planning. Idea Group Publishing, New York (2004)
76. Wezel, W.V., Jorna, R., Meystel, A.: Planning in Intelligent Systems: Aspects, Motivations, and Methods. John Wiley & Sons, Hoboken, New Jersey (2006)
77. Bazan, J.G., Skowron, A., Peters, J.F., Synak, P.: Spatio-temporal approximate reasoning over complex objects. Fundamenta Informaticae 67, 249–269 (2005)
78. Bazan, J.G., Skowron, A., Stepaniuk, J.: Modelling complex patterns by information systems. Fundamenta Informaticae 67, 203–217 (2005)
79. Marcus, S.: The paradox of the heap of grains, in respect to roughness, fuzziness and negligibility. In: Polkowski, L., Skowron, A. (eds.) RSCTC 1998. LNCS, vol. 1424, pp. 19–23. Springer, Heidelberg (1998)
80. Nguyen, S.H., Nguyen, H.S.: Rough set approach to approximation of concepts from taxonomy. In: Proceedings of Knowledge Discovery and Ontologies Workshop (KDO 2004) at ECML/PKDD 2004, Pisa, Italy, University of Pisa, September 24, pp. 13–24 (2004)
81. Nguyen, S.H., Nguyen, T.T., Nguyen, H.S.: Rough set approach to sunspot classification problem. In: Ślęzak, D., et al. (eds.) RSFDGrC 2005. LNCS, vol. 3642, pp. 263–272. Springer, Heidelberg (2005)
82. Orłowska, E.: Semantics of vague concepts. In: Dorn, G., Weingartner, P. (eds.) Foundation of Logic and Linguistics, pp. 465–482. Plenum Press, New York (1984)
83. Skowron, A.: Rough sets and vague concepts. Fundamenta Informaticae 64, 417–431 (2005)

84. Skowron, A., Stepaniuk, J.: Hierarchical modelling in searching for complex patterns: Constrained sums of information systems. Journal of Experimental and Theoretical AI 17, 83–102 (2005)
85. Skowron, A., Synak, P.: Complex patterns in spatio-temporal reasoning. In: Czaja, L. (ed.) Proceedings Workshop on Concurrency, Specification, and Programming (CS&P 2003), Czarna, Poland, September 25-27, vol. 2, pp. 487–499 (2003)
86. Skowron, A., Synak, P.: Complex patterns. Fundamenta Informaticae 60, 351–366 (2004)
87. Zadeh, L.A.: Selected papers. In: Yager, R.R., Ovchinnokov, S., Tong, R., Nguyen, H. (eds.) Fuzzy Sets and Applications: Selected Papers by L.A. Zadeh. John Wiley & Sons, New York (1987)
88. Bazan, J.G.: Rough sets and granular computing in behavioral pattern identification and planning. In: Pedrycz, W., Skowron, A., Kreinovich, V. (eds.) Handbook of Granular Computing, pp. 777–799. John Wiley & Sons, Chichester (2008)
89. Doherty, P., Łukaszewicz, W., Skowron, A., Szałas, A.: Knowledge Engineering: A Rough Set Approach. Springer, Heidelberg (2006)
90. Nguyen, H.S., Skowron, A., Stepaniuk, J.: Granular computing: A rough set approach. Computational Intelligence 17, 514–544 (2001)
91. Skowron, A.: Towards granular multi-agent systems. In: Pal, S.K., Ghosh, A. (eds.) Soft Computing Approach to Pattern Recognition and Image Processing, pp. 215–234. World Scientific, Singapore (2002)
92. Bar-Yam, Y.: Dynamics of Complex Systems. Addison Wesley, New York (1997)
93. Desai, A.: Adaptive complex enterprises. Communications ACM 5, 32–35 (2005)
94. Gell-Mann, M.: The Quark and the Jaguar. Freeman and Co., New York (1994)
95. Luck, M., McBurney, P., Shehory, O., Willmott, S.: Agent technology: Computing as interaction. A roadmap for agent-based computing. Agentlink iii, the european coordination action for agent-based computing, University of Southampton, UK (2005)
96. Sun, R.: Cognition and Multi-Agent Interaction. From Cognitive Modeling to Social Simulation. Cambridge University Press, New York (2006)
97. Urmson, C., et al.: High speed navigation of unrehearsed terrain: Red team technology for grand challenge. Report CMU-RI-TR-04-37, The Robotics Institute, Carnegie Mellon University (2004)
98. Hoen, P.J., Tuyls, K., Panait, L., Luke, S., Poutré, J.A.L.: Overview of cooperative and competitive multiagent learning. In: Tuyls, K., Hoen, P.J., Verbeeck, K., Luke, S. (eds.) LAMAS 2005. LNCS, vol. 3898, pp. 1–46. Springer, Heidelberg (2006)
99. Liu, J.: Autonomous Agents and Multi-Agent Systems: Explorations in Learning, Self-Organization and Adaptive Computation. World Scientific Publishing, Singapore (2001)
100. Liu, J., Jin, X., Tsui, K.: Autonomy Oriented Computing: From Problem Solving to Complex Systems Modeling. Kluwer/Springer, Heidelberg (2005)
101. Luck, M., McBurney, P., Preist, C.: Agent technology: Enabling next generation. a roadmap for agent based computing. Agentlink, University of Southampton, UK (2003)
102. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm intelligence. From natural to artificial systems. In: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI 1999). Oxford University Press, UK (1999)
103. Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Monada, F., Stutzle, T. (eds.): ANTS 2004. LNCS, vol. 3172. Springer, Heidelberg (2004)

104. Peters, J.F.: Rough ethology: Towards a biologically-inspired study of collective behavior in intelligent systems with approximation spaces. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets III. LNCS, vol. 3400, pp. 153–174. Springer, Heidelberg (2005)
105. Gruber, T.R.: A translation approach to portable ontologies. Knowledge Acquisition 5, 199–220 (1993)
106. Smith, B.: Ontology and information systems. Technical report, The Buffalo Ontology Site (2001), `http://ontology.buffalo.edu/ontologyPIC.pdf`
107. W3C: OWL Web Ontology Language, use cases and requirements, W3C recommendation. Technical report, The World Wide Web Consortium Technical Report, `http://www.w3.org/TR/2004/REC-webont-req-20040210/` (2004)
108. Bradbrook, K., Winstanley, G., Glasspool, D., Fox, J., Griffiths, R.: AI planning technology as a component of computerised clinical practice guidelines. In: Miksch, S., Hunter, J., Keravnou, E.T. (eds.) AIME 2005. LNCS, vol. 3581. Springer, Heidelberg (2005)
109. Dojat, M.: Systemes cognitifs pur le traitement d'informations clinques ou biologiques. Habilitation Thesis. L'universite Joseph Fourier, Grenoble, France (1999) (in French)
110. Dojat, M., Pachet, F., Guessoum, Z., Touchard, D., Harf, A., Brochard, L.: Neoganesh: A working system for the automated control of assisted ventilation in icus. Arificial Intelligence in Medicine 11, 97–117 (1997)
111. Glasspool, D., Fox, J., Castillo, F.D., Monaghan, V.E.L.: Interactive decision support for medical planning. In: Dojat, M., Keravnou, E.T., Barahona, P. (eds.) AIME 2003. LNCS, vol. 2780, pp. 335–339. Springer, Heidelberg (2003)
112. Miksch, S.: Plan management in the medical domain. AI Communications 12, 209–235 (1999)
113. Miller, R.: Medical diagnostic decision support systems - past, present, and future. Journal of the American Medical Informatics Association 1, 8–27 (1994)
114. Nilsson, M., Funk, P., Olsson, E.M.G., von Scheele, B., Xiong, N.: Clinical decision-support for diagnosing stress-related disorders by applying psychophysiological medical knowledge to an instance-based learning system. Arificial Intelligence in Medicine 36, 159–176 (2006)
115. OpenClinical, `http://www.openclinical.org`
116. Peek, N.: Decision-theoretic reasoning in medicine: bringing out the diagnosis. In: Proceedings of the Thirteenth Belgian-Dutch Conference on Artificial Intelligence (BNAIC 2001), pp. 203–210 (2001)
117. Sacchi, L., Verduijn, M., Peek, N., de Jonge, E., de Mol, B., Bellazzi, R.: Describing and modeling time series based on qualitative temporal abstraction. In: Peek, N., Combi, C. (eds.) Working notes of the workshop on Intelligent Data Analysis in bioMedicine and Pharmacology (IDAMAP 2006), pp. 31–36 (2006)
118. Shahar, Y., Combi, C.: Temporal reasoning and temporal data maintenance: Issues and challenges. Computers in Biology and Medicine 27, 353–368 (2005)
119. Spyropoulos, C.D.: AI planning and scheduling in the medical hospital environment. Arificial Intelligence in Medicine 20, 101–111 (2000)
120. Stacey, M., McGregor, C.: Temporal abstraction in intelligent clinical data analysis: A survey. Arificial Intelligence in Medicine 39, 1–24 (2007)
121. Verduijn, M., Sacchi, L., Peek, N., Bellazzi, R., de Jonge, E., de Mol, B.: Temporal abstraction for feature extraction: A comparative case study in prediction from intensive care monitoring data. Arificial Intelligence in Medicine 41, 1–12 (2007)
122. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)

123. Peters, J.F.: Approximation spaces in off-policy monte carlo learning. In: Burczynski, T., Cholewa, W., Moczulski, W. (eds.) Recent Methods in Artificial Intelligence Methods. AI-METH Series, Gliwice, Poland, pp. 139–144 (2005)
124. Peters, J.F., Henry, C.: Reinforcement learning with approximation spaces. Fundamenta Informaticae 71, 323–349 (2006)
125. Kleinberg, J., Papadimitriou, C., Raghavan, P.: A microeconomic view of data mining. Data Mining and Knowledge Discovery 2, 311–324 (1998)
126. Keefe, R.: Theories of Vagueness. Cambridge University Press, New York (2000)
127. Keefe, R., Smith, P.: Vagueness: A Reader. MIT Press, Massachusetts (1997)
128. Read, S.: Thinking about Logic: An Introduction to the Philosophy of Logic. Oxford University Press, New York (1994)
129. Zadeh, L.: The concept of linguistic variable and its application to approximate reasoning-I. Information Sciences 8, 199–249 (1975)
130. Zadeh, L.: The concept of linguistic variable and its application to approximate reasoning-II. Information Sciences 8, 301–357 (1975)
131. Zadeh, L.: The concept of linguistic variable and its application to approximate reasoning-III. Information Sciences 9, 43–80 (1975)
132. Dynamics, N.: Special issue on modelling and control of intelligent transportation systems (9 articles). Journal Nonlinear Dynamics 49 (2006)
133. Peters, J.F.: Approximation spaces for hierarchical intelligent behavioral system models. In: Dunin-Keplicz, B., Jankowski, A., Skowron, A., Szczuka, M. (eds.) Monitoring, Security, and Rescue Techniques in Multiagent Systems. Advances in Soft Computing, pp. 13–30. Springer, Heidelberg (2005)
134. Peters, J.F., Henry, C., Ramanna, S.: Rough ethograms: Study of intelligent system behavior. In: Klopotek, M.A., Wierzchon, S.T., Trojanowski, K. (eds.) Proceedings of the International Conference on Intelligent Information Processing and Web Mining (IIS 2005), Gdańsk, Poland, June 13-16, pp. 117–126 (2005)
135. Birattari, M., Caro, G.D., Dorigo, M.: Toward the formal foundation of ant programming. In: Dorigo, M., Di Caro, G.A., Sampels, M. (eds.) Ant Algorithms 2002. LNCS, vol. 2463, pp. 188–201. Springer, Heidelberg (2002)
136. Sukthankar, G., Sycara, K.: A cost minimization approach to human behavior recognition. In: Proceedings of Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (2005)
137. Sukthankar, G., Sycara, K.: Automatic recognition of human team behaviors. In: Proceedings of Modeling Others from Observations (MOO), Workshop at the International Joint Conference on Artificial Intelligence (IJCAI 2005) (July 2005)
138. Adam, N.R., Janeja, V.P., Atluri, V.: Neighborhood based detection of anomalies in high dimensional spatio-temporal sensor datasets. In: Proceedings of the, ACM symposium on Applied computing (SAC 2004), pp. 576–583. ACM Press, New York (2004)
139. Hofmeyr, S.A., Forrest, S., Somayaji, A.: Intrusion detection using sequence of system calls. Journal of Computer Security 2, 151–180 (1998)
140. Lazarevic, A., Ertoz, L., Kumar, V., Ozgur, A., Srivastava, J.: A comparative study of anomaly detection schemes in network intrusion detection. In: Proceedings of the Third SIAM International Conference on Data Mining, San Francisco, CA, USA, May 1-3. SIAM, Philadelphia (2003)
141. Li, X., Han, J., Kim, S., Gonzalez, H.: Roam: Rule- and motif-based anomaly detection in massive moving object data sets. In: Proceedings of the Seventh SIAM International Conference on Data Mining, Minneapolis, Minnesota, USA, April 26-28. SIAM, Philadelphia (2007)

142. Mahoney, M., Chan, P.K.: Learning rules for anomaly detection of hostile network traffic. In: Proceedings of the Third IEEE International Conference on Data Mining (ICDM 2003), Melbourne, Florida, USA, December 19-22, pp. 601–604. IEEE Computer Society, Los Alamitos (2003)
143. Sekar, R., Bendre, M., Dhurjati, D., Bollineni, P.: A fast automaton-based method for detecting anomalous program behaviors. In: Proceedings of the 2001 IEEE Symposium on Security and Privacy (SP 2001), Washington, DC, USA, pp. 144–155. IEEE Computer Society Press, Los Alamitos (2001)
144. Shavlik, J.W., Shavlik, M.: Selection, combination, and evaluation of effective software sensors for detecting abnormal computer usage. In: Kim, W., Kohavi, R., Gehrke, J., DuMouchel, W. (eds.) Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, USA, August 22-25, pp. 276–285 (2004)
145. Flach, P.A., Lachiche, N.: Naive bayesian classification of structured data. Machine Learning 57, 233–269 (2004)
146. Grandidier, F., Sabourin, R., Suen, C.Y.: Integration of contextual information in handwriting recognition systems. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003), Washington, DC, USA, p. 1252. IEEE Computer Society Press, Los Alamitos (2003)
147. Guo, G., Li, S., Chan, K.: Face recognition by support vector machines. In: Proceedings of the International Conferences on Automatic Face and Gesture Recognition, pp. 196–201 (2000)
148. Li, Z.C., Suen, C.Y., Guo, J.: Analysis and recognition of alphanumeric handprints by parts. In: Proceedings of the ICPR 1992, pp. 338–341 (1992)
149. Li, Z.C., Suen, C.Y., Guo, J.: A regional decomposition method for recognizing handprinted characters. SMC 25, 998–1010 (1995)
150. Nguyen, T.T.: Understanding domain knowledge: Concept approximation using rough mereology. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2005), Washington, DC, USA, pp. 217–222. IEEE Computer Society, Los Alamitos (2005)
151. Pavlidis, T.: Structural Pattern Recognition. Springer, Heidelberg (1980)
152. Yousfi, K., Ambroise, C., Cocquerez, J.P., Chevelu, J.: Supervised learning for guiding hierarchy construction: Application to osteo-articular medical images database. In: Proceedings of the 18th International Conference on Pattern Recognition (ICPR 2006), Washington, DC, USA, pp. 484–487. IEEE Computer Society Press, Los Alamitos (2006)
153. WITAS, `http://www.ida.liu.se/ext/witas/eng.html`
154. Hoare, C.A.R.: Process algebra: A unifying approach. In: Abdallah, A.E., Jones, C.B., Sanders, J.W. (eds.) Communicating Sequential Processes. LNCS, vol. 3525. Springer, Heidelberg (2005)
155. Jensen, K.: Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Springer, New York (2006)
156. Roscoe, A.W., Hoare, C.A.R., Bird, R.: The Theory and Practice of Concurrency. Prentice Hall, New York (1997)
157. Suraj, Z.: Tools for generating concurrent models specified by information systems. In: Lin, T.Y., Wildberger, A.M. (eds.) Soft Computing: Rough Sets, Fuzzy Logic, Neural Networks, Uncertainty Management, Knowledge Discovery, pp. 107–110. Society For Computer Simulation, San Diego (1995)
158. Taubenfeld, G.: Synchronization Algorithms and Concurrent Programming. Prentice-Hall, New York (2006)

159. Winskel, G., Nielsen, M.: Models for concurrency. In: Handbook of logic in computer science: semantic modelling, vol. 4, pp. 1–148. Oxford University Press, Oxford (1995)
160. Kauppinen, T., Hyvonen, E.: Modeling and reasoning about changes in ontology time series. In: Kishore, R., Ramesh, R., Sharman, R. (eds.) Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems, pp. 319–338. Springer, New York (2007)
161. Voigt, K.: Reasoning about changes and uncertainty in browser customization. In: Proceedings of the AAAI Fall Symposium, Working Notes of AI Applications to Knowledge Navigation and Retrieval (1995)
162. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning a mahalanobis metric from equivalence constraints. Journal of Machine Learning Research 6, 937–965 (2005)
163. Dzeroski, S., Lavrac, N. (eds.): Relational Data Mining. Springer, New York (2001)
164. Gartner, T.: A survey of kernels for structured data. SIGKDD Explorations 5, 48–58 (2003)
165. Nguyen, S.H.: Regularity Analysis and Its Applications in Data Mining. PhD thesis, Warsaw University, Warsaw, Poland (2000)
166. Shalev-Shwartz, S., Singer, Y., Ng, A.Y.: Online and batch learning of pseudo-metrics. In: Proceedings of the twenty-first international conference on Machine learning (ICML 2004), vol. 4585. ACM Press, New York (2004)
167. Weinberger, K., Blitzer, J., Saul, L. (eds.): Distance metric learning for large margin nearest neighbor classification. Advances in Neural Information Processing Systems, vol. 18. MIT Press, Cambridge (2006)
168. Wettschereck, D., Aha, D.W., Mohri, T.: A review and empirical evaluation of feature weighting methods for aclass of lazy learning algorithms. Artifficial Intelligence Review 11, 273–314 (1997)
169. Wojna, A.: Analogy-based Reasoning in Classifier Construction. PhD thesis, Warsaw University; Faculty of Mathematics, Informatics and Mechanics, Warsaw, Poland (2004); Defense in 2005. Awarded with honours
170. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S. (eds.): Distance metric learning with application to clustering with side-information. Advances in NIPS, vol. 15. MIT Press, Cambridge (2003)
171. Yang, L., Jin, R., Sukthankar, R., Liu, Y.: An efficient algorithm for local distance metric learning. In: Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI 2006). ACM Press, New York (2006)
172. Bazan, J.G., Skowron, A.: Classifiers based on approximate reasoning schemes. In: Dunin-Kęplicz, B., Jankowski, A., Skowron, A., Szczuka, M. (eds.) Monitoring, Security, and Rescue Techniques in Multiagent Systems. Advances in Soft Computing, pp. 191–202. Springer, Heidelberg (2005)
173. Bazan, J.G.: Behavioral pattern identification through rough set modeling. Fundamenta Informaticae 72, 37–50 (2006)
174. Bazan, J.G., Kruczek, P., Bazan-Socha, S., Skowron, A., Pietrzyk, J.: Risk pattern identification in the treatment of infants with respiratory failure through rough set modeling. In: Proceedings of the Eleventh Conference of Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2006), July 2-7, Paris, France, pp. 2650–2657 (2006)
175. Bazan, J.G., Kruczek, P., Bazan-Socha, S., Skowron, A., Pietrzyk, J.: Rough set approach to behavioral pattern identification. Fundamenta Informaticae 75, 27–47 (2007)

176. Bazan, J.G., Peters, J.F., Skowron, A.: Behavioral pattern identification through rough set modelling. In: Ślęzak, D., et al. (eds.) RSFDGrC 2005. LNCS, vol. 3642, pp. 688–697. Springer, Heidelberg (2005)
177. Bazan, J.G., Skowron, A.: On-line elimination of non-relevant parts of complex objects in behavioral pattern identification. In: Pal, S.K., et al. (eds.) PReMI 2005. LNCS, vol. 3776, pp. 720–725. Springer, Heidelberg (2005)
178. Bazan, J.G., Skowron, A.: Perception based rough set tools in behavioral pattern identification. In: Czaja, L. (ed.) Proceedings Workshop on Concurrency, Specification, and Programming (CS&P 2005), September 28-30, pp. 50–56. Warsaw University, Warsaw (2005)
179. Nguyen, S.H., Bazan, J., Skowron, A., Nguyen, H.S.: Layered learning for concept synthesis. LNCS Transactions on Rough Sets 3100, 187–208 (2004)
180. Bazan, J.G., Kruczek, P., Bazan-Socha, S., Skowron, A., Pietrzyk, J.: Automatic planning based on rough set tools: Towards supporting treatment of infants with respiratory failure. In: Proceedings of the Workshop on Concurrency, Specification, and Programming (CS&P 2006), Wandlitz, Germany, September 27-29. Informatik-Bericht, vol. 170, pp. 388–399. Humboldt University, Berlin (2006)
181. Bazan, J.G., Kruczek, P., Bazan-Socha, S., Skowron, A., Pietrzyk, J.: Automatic planning of treatment of infants with respiratory failure through rough set modeling. In: Greco, S., Hata, Y., Hirano, S., Inuiguchi, M., Miyamoto, S., Nguyen, H.S., Słowiński, R. (eds.) RSCTC 2006. LNCS, vol. 4259, pp. 418–427. Springer, Heidelberg (2006)
182. Stone, P.: Layered Learning in Multi-Agent Systems: A Winning Approach to Robotic Soccer. The MIT Press, Cambridge (2000)
183. Bolc, L., Szałas, A. (eds.): Time and Logic: A Computational Approach. UCL Press, London (1995)
184. Clark, E., Emerson, E., Sistla, A.: Automatic verification of finite state concurrent systems using temporal logic specifications: A practical approach. ACM Transactions on Programming Languages and Systems 8, 244–263 (1986)
185. Emerson, E.A.: Temporal and modal logic. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science: Volume B: Formal Models and Semantics, pp. 995–1072. Elsevier, Amsterdam (1990)
186. Skowron, A., Stepaniuk, J.: Constrained sums of information systems. In: Tsumoto, S., Słowiński, R., Komorowski, J., Grzymała-Busse, J.W. (eds.) RSCTC 2004. LNCS, vol. 3066, pp. 300–309. Springer, Heidelberg (2004)
187. Skowron, A., Stepaniuk, J.: Ontological framework for approximation. In: Ślęzak, D., et al. (eds.) RSFDGrC 2005. LNCS, vol. 3641, pp. 718–727. Springer, Heidelberg (2005)
188. Komorowski, J., Polkowski, L., Skowron, A.: Towards a rough mereology-based logic for approximate solution synthesis. Special Issue on Logics with Incomplete Information, Studia Logica 58, 143–184 (1997)
189. Komorowski, J., Polkowski, L., Skowron, A.: Rough sets: a tutorial. In: Pal, S.K., Skowron, A. (eds.) Rough Fuzzy Hybridization: A New Trend in Decision-Making, pp. 3–98. Springer, Singapore (1999)
190. Polkowski, L., Skowron, A.: Rough mereology. In: Raś, Z.W., Zemankova, M. (eds.) ISMIS 1994. LNCS, vol. 869, pp. 85–94. Springer, Heidelberg (1994)
191. Polkowski, L., Skowron, A.: Rough mereological approach to knowledge-based distributed ai. In: Lee, J.K., Liebowitz, J., Chae, Y.M. (eds.) Proceedings of the Third World Congress on Expert Systems, February 5-9, Soeul, Korea, pp. 774–781. Cognizant Communication Corporation (1996)

192. Polkowski, L., Skowron, A.: Rough mereology: A new paradigm for approximate reasoning. International Journal of Approximate Reasoning 15, 333–365 (1996)
193. Polkowski, L., Skowron, A.: Rough mereology in information systems. a case study: Qualitative spatial reasoning. In: Polkowski, L., Lin, T.Y., Tsumoto, S. (eds.) Rough Set Methods and Applications: New Developments in Knowledge Discovery in Information Systems. Studies in Fuzziness and Soft Computing, vol. 56. Springer, Heidelberg (2000)
194. Polkowski, L., Skowron, A.: Rough mereological calculi of granules: A rough set approach to computation. Computational Intelligence 17, 472–492 (2001)
195. Bazan, J.G.: Dynamic reducts and statistical inference. In: Proceedings of the Sixth International Conference on Information Processing and Management of Uncertainty on Knowledge Based Systems (IPMU 1996), Granada, Spain, July 1-5, vol. III, pp. 1147–1152 (1996)
196. Bazan, J.G.: A comparison of dynamic and non-dynamic rough set methods for extracting laws from decision tables. In: Polkowski, L., Skowron, A. (eds.) Rough Sets in Knowledge Discovery 1: Methodology and Applications. Studies in Fuzziness and Soft Computing, vol. 18, pp. 321–365. Physica-Verlag, Heidelberg (1998)
197. Bazan, J.G.: Discovery of decision rules by matching new objects against data tables. In: Polkowski, L., Skowron, A. (eds.) RSCTC 1998. LNCS (LNAI), vol. 1424, pp. 521–528. Springer, Heidelberg (1998)
198. Bazan, J.G.: Methods of approximate reasoning for synthesis of decision algorithms. PhD thesis, Warsaw University, Warsaw, Poland (1999) (in Polish)
199. Bazan, J.G., Nguyen, H.S., Nguyen, S.H., Synak, P., Wróblewski, J.: Rough set algorithms in classification problems. In: Polkowski, L., Lin, T.Y., Tsumoto, S. (eds.) Rough Set Methods and Applications: New Developments in Knowledge Discovery in Information Systems. Studies in Fuzziness and Soft Computing, vol. 56, pp. 49–88. Springer, Heidelberg (2000)
200. Bazan, J.G., Nguyen, H.S., Szczuka, M.: A view on rough set concept approximations. Fundamenta Informaticae 59, 107–118 (2004)
201. Bazan, J.G., Skowron, A., Synak, P.: Discovery of decision rules from experimental data. In: Lin, T.Y. (ed.) Proceedings of the Third International Workshop on Rough Sets and Soft Computing (RSSC 1994), San Jose, CA, USA, pp. 526–533 (1994)
202. Bazan, J.G., Skowron, A., Synak, P.: Dynamic reducts as a tool for extracting laws from decision tables. In: Raś, Z.W., Zemankova, M. (eds.) ISMIS 1994. LNCS (LNAI), vol. 869, pp. 346–355. Springer, Heidelberg (1994)
203. Bazan, J.G., Skowron, A., Synak, P.: Market data analysis: A rough set approach. ICS Research Reports 6, Warsaw University of Technology, Warsaw, Poland (1994)
204. Barwise, J., Seligman, J.: The logic of distributed systems. Cambridge University Press, Cambridge (1997)
205. Fahlman, S.E., Lebiere, C.: The Cascade-Correlation Learning Architecture. Advances in Neural Information Processing Systems, vol. II. Morgan Kaufmann, San Mateo (1990)
206. Cedrowska, J.: Prism: An algorithm for inducing modular rules. In: Gaines, B.R., Boose, J.H. (eds.) Knowledge Acquisition for Knowledge-based Systems. Academic Press, New York (1988)
207. Cestnik, B., Kononenko, I., Bratko, I.: Assistant 86: A knowledge elicitation tool for sophisticated users. In: Proceedings of EWSL 1987, Bled, Yugoslavia, pp. 31–47 (1997)

208. Goodman, R.M., Smyth, P.: The induction of probabilistic rule sets – the algorithm. In: Proceedings of the Sixth International Workshop on Machine Learning, San Mateo, CA, pp. 129–132. Morgan Kaufmann, San Francisco (1989)
209. Quinlan, J.R.: The cascade-correlation learning architecture. In: Machine Learning 1, pp. 81–106. Kluwer Academic, Boston (1990)
210. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo (1993)
211. Utgoff, P.E.: Incremental learning of decision trees. In: Machine Learning 1, vol. IV, pp. 161–186. Kluwer Academic, Boston (1990)
212. Velde, W.V.D.: Idl, or taming the multiplexer. In: Proceedings of the Fourth European Working Session on Learning, Pitman, London, pp. 31–47 (1989)
213. Bloedorn, E., Michalski, R.S.: The multi-purpose incremental learning system AQ15 and its testing to three medical domains. In: Proceedings of the Third International Conference on Tools for AI, San Jose, CA (1991)
214. Clark, P., Niblett, T.: The CN2 induction algorithm. In: Machine Learning 3, pp. 261–284. Kluwer Academic, Boston (1989)
215. Grzymała-Busse, J.W.: LERS - a system for learning from examples based on rough sets. In: Słowiński, R. (ed.) Intelligent Decision Support - Handbook of Applications and Advances of the Rough Sets Theory. D: System Theory, Knowledge Engineering and Problem Solving, vol. 11, pp. 3–18. Kluwer Academic Publishers, Dordrecht (1992)
216. Grzymała-Busse, J.W.: LERS - a knowledge discovery system. In: Polkowski, L., Skowron, A. (eds.) Rough Sets in Knowledge Discovery 2. Applications, Case Studies and Software Systems. Studies in Fuzziness and Soft Computing, pp. 562–565. Physica-Verlag, Heidelberg (1998)
217. Michalski, R.S., Mozetic, I., Hong, J., Lavrac, N.: The multi-purpose incremental learning system AQ15 and its testing to three medical domains. In: Proceedings of the AAAI 1986, San Mateo, CA, pp. 1041–1045. Morgan Kaufmann, San Francisco (1986)
218. Michalski, R.S., Wnęk, J.: Constructive induction: An automated improvement of knowledge representation spaces for machine learning. In: Proceedings of a Workshop on Intelligent Information Systems, Practical Aspect of AI II, Augustów, Poland, pp. 188–236. Morgan Kaufmann, San Francisco (1993)
219. Mienko, R., Słowiński, R., Stefanowski, J., Susmaga, R.: Roughfamily - software implementation of rough set based data analysis and rule discovery techniques. In: Tsumoto, S. (ed.) Proceedings of the Fourth International Workshop on Rough Sets, Fuzzy Sets and Machine Discovery, Tokyo, Japan, November 6-8, pp. 437–440 (1996)
220. Øhrn, A., Komorowski, J.: ROSETTA - a rough set tool kit for analysis of data. In: Tsumoto, S. (ed.) Proceedings of the Fifth International Workshop on Rough Sets and Soft Computing (RSSC 1997) at the Third Joint Conference on Information Sciences (JCIS 1997), Research Triangle Park, NC, USA, March 2-5, pp. 403–407 (1997)
221. Słowiński, R., Stefanowski, J.: 'RoughDAS' and 'RoughClass' software implementations of the rough set approach. In: Słowiński, R. (ed.) Intelligent Decision Support - Handbook of Applications and Advances of the Rough Sets Theory. D: System Theory, Knowledge Engineering and Problem Solving, vol. 11. Kluwer Academic Publishers, Dordrecht (1992)

222. Stefanowski, J.: On rough set based approaches to induction of decision rules. In: Polkowski, L., Skowron, A. (eds.) Rough Sets in Knowledge Discovery 1: Methodology and Applications. Studies in Fuzziness and Soft Computing, vol. 18, pp. 500–529. Physica-Verlag, Heidelberg (1998)
223. Stefanowski, J.: Algorithims of rule induction for knowledge discovery. Habilitation Thesis published as Series Rozprawy no. 361. Poznań University of Technology Press, Poznań (2001) (in Polish)
224. Dzeroski, S.: Handling noise in inductive logic programming. Master's thesis, Dept. of EE and CS, University of Ljubljana, Ljubljana, Slovenia (1991)
225. Bazan, J.G., Latkowski, R., Szczuka, M.: DIXER - distributed executor for rough set exploration system. In: Ślęzak, D., et al. (eds.) RSFDGrC 2005. LNCS, vol. 3642, pp. 39–47. Springer, Heidelberg (2005)
226. Bazan, J.G., Szczuka, M.: RSES and RSESlib – a collection of tools for rough set computations. In: Ziarko, W., Yao, Y. (eds.) RSCTC 2000. LNCS (LNAI), vol. 2005, pp. 106–113. Springer, Heidelberg (2001)
227. Bazan, J.G., Szczuka, M., Wojna, A., Wojnarski, M.: On the evolution of rough set exploration system. In: Tsumoto, S., Słowiński, R., Komorowski, J., Grzymała-Busse, J.W. (eds.) RSCTC 2004. LNCS, vol. 3066, pp. 592–601. Springer, Heidelberg (2004)
228. Bazan, J.G., Szczuka, M.S., Wróblewski, J.: A new version of rough set exploration system. In: Alpigini, J.J., Peters, J.F., Skowron, A., Zhong, N. (eds.) RSCTC 2002. LNCS, vol. 2475, pp. 397–404. Springer, Heidelberg (2002)
229. Synak, P.: Rough Set Expert System, User's Guide. Technical report, Warsaw University, Warsaw, Poland (1995)
230. Synak, P., Bazan, J.G., Cykier, A.: RSES Core Classes: The Technical Documentation. Technical report, Warsaw University, Warsaw, Poland (1996)
231. Szczuka, M.: Mikołajczyk, M., Bazan, J.G.: RSES 2.2 user's guide. Technical report, Warsaw University, Warsaw, Poland (2005)
232. Polkowski, L.: Rough Sets: Mathematical Foundations. Advances in Soft Computing. Springer-Verlag/Physica-Verlag, Heidelberg (2002)
233. Lipski, W.: On databases with incomplete information. Journal of the ACM 28, 41–70 (1981)
234. Pawlak, Z.: Classification of objects by means of attributes. Polish Academy of Sciences 429 (1981)
235. Orłowska, E., Pawlak, Z.: Representation of nondeterministic information. Theoretical Computer Science 29, 27–39 (1984)
236. Orłowska, E. (ed.): Incomplete Information: Rough Set Analysis. Studies in Fuzziness and Soft Computing, vol. 13. Physica-Verlag, Heidelberg (1998)
237. Nguyen, H.S.: Discretization of Real Value Attributes, Boolean Reasoning Approach. Ph.D thesis, Warsaw University, Warsaw, Poland (1997)
238. Nguyen, H.S.: Approximate boolean reasoning: Foundations and applications in data mining. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets V. LNCS, vol. 4100, pp. 334–506. Springer, Heidelberg (2006)
239. Polkowski, L., Skowron, A.: Synthesis of decision systems from data tables. In: Lin, T.Y., Cecerone, N. (eds.) Rough Sets and Data Mining. Analysis for Imprecise Data. Advances in Soft Computing, pp. 259–299. Kluwer Academic, Dordrecht (1997)
240. Pawlak, Z., Skowron, A.: A rough set approach for decision rules generation. In: ICS Research Report 23/93, Warsaw University of Technology and Proceedings of the IJCAI 1993 Workshop W12: The Management of Uncertainty in AI, France (1993)

241. Tsumoto, S., Tanaka, H.: PRIMEROSE: Probabilistic rule induction method based on rough sets and resampling methods. Computational Intelligence 11, 389–405 (1995)
242. Synak, P.: Methods of approximate reasoning in searching for rough dependencies. Master's thesis, Warsaw University, Warsaw, Poland (1996) (in Polish)
243. Mitchell, T.M.: Machine Learning. McGraw-Hill, New York (1997)
244. Bennett, B.: The role of definitions in construction and analysis of formal ontologies. In: Sixth Symposium on Logical Formalizations of Commonsense Reasoning, Palo Alto, CA, USA (2003)
245. Tarski, A.: Some methodological investigations on the definability of concepts. In: Logic, Semantics, Metamathematics. Clarendon Press, Oxford (1956)
246. Bazan, J.G., Skowron, A., Świniarski, R.W.: Rough sets and vague concept approximation: From sample approximation to adaptive learning. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets V. LNCS, vol. 4100, pp. 39–62. Springer, Heidelberg (2006)
247. Skowron, A., Stepaniuk, J., Peters, J.F., Świniarski, R.W.: Calculi of approximation spaces. Fundamenta Informaticae 72, 363–378 (2006)
248. Skowron, A., Świniarski, R.: Rough sets and higher order vagueness. In: Ślęzak, D., et al. (eds.) RSFDGrC 2005. LNCS, vol. 3641, pp. 33–42. Springer, Heidelberg (2005)
249. Skowron, A., Świniarski, R.W., Synak, P.: Approximation spaces and information granulation. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets III. LNCS, vol. 3400, pp. 175–189. Springer, Heidelberg (2005)
250. Zadeh, L.: Fuzzy sets. Information and Control, 338–353 (1965)
251. Ajdukiewicz, K.: Pragmatic Logic. Reidel, Dordrecht (1974)
252. Holland, J.H., Holyoak, K.J., Nisbett, R.E., Thagard, P.R.: Induction: processes of inference, learning, and discovery. MIT Press, Cambridge (1989)
253. Skowron, A., Stepaniuk, J.: Tolerance approximation spaces. Fundamenta Informaticae 27, 245–253 (1996)
254. Pawlak, Z., Skowron, A.: Rough sets: Some extensions. Information Sciences 177, 28–40 (2007)
255. Stepaniuk, J.: Knowledge discovery by application of rough set models. In: Polkowski, L., Lin, T.Y., Tsumoto, S. (eds.) FCT 1977. Studies in Fuzziness and Soft Computing, vol. 56, pp. 137–233. Springer-Verlag/Physica-Verlag, Heidelberg (2000)
256. Ziarko, W.: Variable precision rough set model. Journal of Computer and System Sciences 46, 39–59 (1993)
257. Provost, F., Kohavi, R.: On applied research in machine learning. Machine Learning 30, 127–132 (1998)
258. Weiss, S.M., Kulikowski, C.A.: Computer Systems That Learn. Morgan Kaufmann, San Mateo (1991)
259. ROSETTA: Project web site, `http://rosetta.lcb.uu.se/general`
260. Altman, D.G.: Practical Statistics for Medical Research. Chapman and Hall/CRC, London (1997)
261. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters 27, 861–874 (2006)
262. Swets, J.A.: Measuring the accuracy of diagnostic systems. Science 240, 1285–1293 (1988)
263. Øhrn, A., Komorowski, J., Skowron, A., Synak, P.: The design and implementation of a knowledge discovery toolkit based on rough sets: The ROSETTA system. In: Polkowski, L., Skowron, A. (eds.) Rough Sets in Knowledge Discovery 1: Methodology and Applications. Studies in Fuzziness and Soft Computing, vol. 18, pp. 376–399. Physica-Verlag, Heidelberg (1998)

264. Øhrn, A., Komorowski, J., Skowron, A., Synak, P.: The ROSETTA software system. In: Polkowski, L., Skowron, A. (eds.) Rough Sets in Knowledge Discovery 2. Applications, Case Studies and Software Systems. Studies in Fuzziness and Soft Computing, pp. 572–576. Physica-Verlag, Heidelberg (1998)
265. Efron, B.: Estimating the error rate of a prediction rule: improvement on cross validation. Journal of American Statistics Association 78, 316–331 (1983)
266. Stefanowski, J.: Classification and decision supporting based on rough set theory. Foundations of Computing and Decision Sciences 18, 371–380 (1993)
267. Delimata, P., Moshkov, M., Skowron, A., Suraj, Z.: Two families of classification algorithms. In: An, A., Stefanowski, J., Ramanna, S., Butz, C.J., Pedrycz, W., Wang, G. (eds.) RSFDGrC 2007. LNCS, vol. 4482, pp. 297–304. Springer, Heidelberg (2007)
268. Bazan, J.G., Nguyen, H.S., Skowron, A., Szczuka, M.S.: A view on rough set concept approximations. In: Wang, G., Liu, Q., Yao, Y., Skowron, A. (eds.) RSFDGrC 2003. LNCS, vol. 2639, pp. 181–188. Springer, Heidelberg (2003)
269. Skowron, A., Peters, J.F.: Rough sets: Trends and challenges - plenary paper. In: Wang, G., Liu, Q., Yao, Y., Skowron, A. (eds.) RSFDGrC 2003. LNCS, vol. 2639, pp. 25–34. Springer, Heidelberg (2003)
270. Webster: Webster's New Collegiate Dictionary. Merriam-Webster, Springfield (1991)
271. Liliana, A.: Material and formal ontology. In: Roberto, P., Peter, S. (eds.) Formal ontology. Advanced in Soft Computing, pp. 199–232. Kluwer, Dordrecht (1996)
272. Guarino, N., Poli, R.: Formal ontology in conceptual analysis and knowledge representation. Kluwer, Dordrecht (1993)
273. Shapiro, S.C.: Encyclopedia of Artificial Intelligence. John Wiley and Sons, New York (1992)
274. Booch, G.: Object-oriented Analysis and Design with Applications. Addison-Wesley Publishing Company, Santa Clara (1994)
275. Taylor, D.A.: Object-Oriented Information Systems: Planning and Implementation. John Wiley & Sons, New York (1992)
276. Jones, D., Bench-Capon, T., Visser, P.: Ontology-based support for human disease study. In: Proceedings of the IT&KNOWS Conference, XV IFIP World Computer Congress (August 1998)
277. Uschold, M., Grüninger, M.: Ontologies: principles, methods, and applications. Knowledge Engineering Review 11, 93–155 (1996)
278. Uschold, M.: Building ontologies: Towards a unified methodology. In: Proceedings 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems, Cambridge, UK (1996)
279. Dublin Core: Project web site, `http://dublincore.org/`
280. General Formal Ontology (GFO): Project web site, `http://www.onto-med.de/`
281. OpenCyc/ResearchCyc: Project web site, `http://research.cyc.com/`
282. Suggested Upper Merged Ontology (SUMO): Project web site, `http://www.articulatesoftware.com/`
283. WordNet: Project web site, `http://wordnet.princeton.edu/`
284. Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE): Project web site, `http://www.loa-cnr.it/DOLCE.html`
285. W3C: RDF Primer, W3C Recommendation. Technical report, The World Wide Web Consortium Technical Report (2004), `http://www.w3.org/RDF/`
286. OIL - Ontology Inference Layer: Project web site, `www.ontoknowledge.org/oil/`
287. DAML – DARPA Agent Markup Language: Project web site, `www.daml.org`

288. Cyc: Project web site, `http://www.cyc.com`
289. OpenCyc: Project web site, `http://opencyc.org`
290. Protege: Project web site, `http://protege.stanford.edu`
291. OntoStudio: Project web site, `http://www.ontoprise.de`
292. Ontolingua: Project web site, `www.ksl.stanford.edu/software/ontolingua/`
293. Chimaera: Project web site, `http://ksl.stanford.edu/software/chimaera/`
294. OilEd: Project web site, `http://oiled.man.ac.uk/`
295. W3C: RDQL - a query language for RDF, W3C member submission. Technical report, The World Wide Web Consortium Technical Report (2004), `http://www.w3.org/Submission/RDQL`
296. Fahle, M., Poggio, T.: Perceptual Learning. MIT Press, Cambridge (2002)
297. Harnad, S.: Categorical Perception: The Groundwork of Cognition. Cambridge University Press, New York (1987)
298. McCarthy, J.: Notes on formalizing context. In: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI 1993). Morgan Kaufmann, Chambéry (1993)
299. McCarthy, J., Hayes, P.: Some philosophical problems from the standpoint of artificial intelligence. In: Meltzer, B., Michie, D. (eds.) Machine Intelligence, vol. IV, pp. 463–502. Edinburgh University Press, Edinburgh (1969)
300. Anderson, J.R.: Rules of the mind. Lawrence Erlbaum, Hillsdale (1993)
301. Kieras, D., Meyer, D.E.: An overview of the epic architecture for cognition and performance with application to human-computer interaction. Human-Computer Interaction 12, 391–438 (1997)
302. Laird, J., Newell, A., Rosenbloom, P.: Soar: An architecture for general intelligence. Artificial Intelligence 33, 1–64 (1987)
303. Veloso, M.M., Carbonell, J.G.: Derivational analogy in prodigy: Automating case acquisition, storage, and utilization. Machine Learning 10, 249–278 (1993)
304. Tarski, A.: The semantic concept of truth. Philosophy and Phenomenological Research 4, 341–375 (1944)
305. Dictionary, T.F.: Project web site, `http://www.thefreedictionary.com`
306. Bazan, J.G., Nguyen, S.H., Nguyen, H.S., Skowron, A.: Rough set methods in approximation of hierarchical concepts. In: Tsumoto, S., Słowiński, R., Komorowski, J., Grzymała-Busse, J.W. (eds.) RSCTC 2004. LNCS, vol. 3066, pp. 346–355. Springer, Heidelberg (2004)
307. Nguyen, S.H., Nguyen, H.S.: Improving Rough Classifiers Using Concept Ontology. In: Ho, T.B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS, vol. 3518, pp. 312–322. Springer, Heidelberg (2005)
308. Greco, S., Matarazzo, B., Słowiński, R.: A new rough set approach to multicriteria and multiattribute classification. In: Polkowski, L., Skowron, A. (eds.) RSCTC 1998. LNCS (LNAI), vol. 1424, pp. 60–67. Springer, Heidelberg (1998)
309. Greco, S., Matarazzo, B., Słowiński, R.: Rough approximation of preference relation by dominance relations. ICS Research Report 16/1996, Warsaw University of Technology, Warsaw, Poland; also in Journal of Operational Research 117, 63–83 (1999)
310. Greco, S., Matarazzo, B., Słowiński, R.: Multicriteria classification. In: Kloesgen, W., Zytkow, J. (eds.) Handbook of Data Mining and Knowledge Discovery, pp. 318–328. Oxford University Press, Inc., New York (2002)
311. Greco, S., Matarazzo, B., Słowiński, R.: Rough approximation by dominance relations. International Journal of Intelligent Systems 17, 153–171 (2002)

312. Błaszczyński, J., Greco, S., Słowiński, R.: Multi-criteria classification – a new scheme for application of dominance-based decision rules. Journal of Operational Research 181, 1030–1044 (2007)
313. Błaszczyński, J., Słowiński, R.: Incremental induction of decision rules from dominance-based rough approximations. In: Skowron, A., Szczuka, M. (eds.) Electronic Notes in Theoretical Computer Science, vol. 82. Springer, Heidelberg (2003)
314. Błaszczyński, J., Słowiński, R.: Incremental induction of satisfactory decision rules from dominance based rough approximations. In: Skowron, A., Szczuka, M. (eds.) Proceedings of the International Workshop on Rough Sets in Knowledge Discovery and Soft Computing (RSKD 2003), Warsaw, Poland, April 12-13, pp. 40–51 (2003)
315. Greco, S., Matarazzo, B., Słowiński, R., Stefanowski, J.: An algorithm for induction of decision rules consistent with the dominance principle. In: Ziarko, W.P., Yao, Y. (eds.) RSCTC 2000. LNCS, vol. 2005, pp. 304–313. Springer, Heidelberg (2001)
316. Peters, J.F.: Time and clock information systems: Concepts and rough fuzzy petri net models. In: Polkowski, L., Skowron, A. (eds.) Rough Sets in Knowledge Discovery 2. Applications, Case Studies and Software Systems. Studies in Fuzziness and Soft Computing, pp. 385–417. Springer-Verlag, Berlin (1998)
317. Polkowski, L.: Granulation of knowledge in decision systems: The approach based on rough inclusions. The method and its applications. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS, vol. 4585, pp. 69–79. Springer, Heidelberg (2007)
318. Synak, P.: Temporal Aspects of Data Analysis: A Rough Set Approach. Ph.D thesis, The Institute of Computer Science of the Polish Academy of Sciences, Warsaw, Poland (2003) (in Polish) (defended in 2004)
319. Polkowski, L., Artiemjew, P.: On granular rough computing with missing values. In: Kryszkiewicz, M., Peters, J.F., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS, vol. 4585, pp. 271–279. Springer, Heidelberg (2007)
320. Skowron, A.: Toward intelligent systems: Calculi of information granules. In: Terano, T., Nishida, T., Namatame, A., Tsumoto, S., Ohsawa, Y., Washio, T. (eds.) JSAI-WS 2001. LNCS (LNAI), vol. 2253, pp. 251–260. Springer, Heidelberg (2001)
321. Stepaniuk, J.: Approximation spaces, reducts and representatives. In: Polkowski, L., Skowron, A. (eds.) Rough Sets in Knowledge Discovery 1: Methodology and Applications. Studies in Fuzziness and Soft Computing, vol. 18, pp. 109–126. Physica-Verlag, Heidelberg (1998)
322. Stepaniuk, J.: Knowledge discovery by application of rough set models. In: Polkowski, L., Lin, T.Y., Tsumoto, S. (eds.) Rough Set Methods and Applications: New Developments in Knowledge Discovery in Information Systems. Studies in Fuzziness and Soft Computing, vol. 56, pp. 137–233. Physica-Verlag, Heidelberg (2000)
323. Yao, Y.Y.: Perspectives of granular computing. In: Proceedings of the Conference on Granular Computing (GrC 2005), Beijing, China, New York. IEEE Press, Los Alamitos (2005)
324. Bazan, J.G., Osmólski, A., Skowron, A., Ślęzak, D., Szczuka, M., Wróblewski, J.: Rough set approach to survival analysis. In: Suraj, Z. (ed.) Proceedings of the Sixth International Conference on Soft Computing and Distributed Processing (SCDP 2002), June 24-25, pp. 45–48. University of Information Technology and Management in Rzeszów Press, Rzeszów (2002)

325. Bazan, J.G., Osmólski, A., Skowron, A., Ślęzak, D., Szczuka, M., Wróblewski, J.: Rough set approach to the survival analysis. In: Alpigini, J.J., Peters, J.F., Skowron, A., Zhong, N. (eds.) RSCTC 2002. LNCS, vol. 2475, pp. 522–529. Springer, Heidelberg (2002)
326. Bazan, J.G., Skowron, A., Ślęzak, D., Wróblewski, J.: Searching for the complex decision reducts: The case study of the survival analysis. In: Raś, Z.W., Zhong, N., Tsumoto, S., Suzuku, E. (eds.) ISMIS 2003. LNCS, vol. 2871, pp. 160–168. Springer, Heidelberg (2003)
327. Vincent, J.L., de Mendonca, A., Cantraine, F., et al.: Use of the sofa score to assess the incidence of organ dysfunction. Crit Care Medicine 26, 1793–1800 (1998)
328. Hurford, W.E., Bigatello, L.M., Haspel, K.L., Hess, D.R., Warren, R.L.: Critical care handbook of the Massachusetts General Hospital, 3rd edn. Lippincott Williams & Wilkins, Philadelphia (2000)
329. Revelation Software: Web site, `http://www.revelation.com/`
330. Ginsberg, M.L.: Approximate planning. Artificial Intelligence 76, 89–123 (1995)
331. Ginsberg, M.L.: A new algorithm for generative planning. In: Aiello, L.C., Doyle, J., Shapiro, S. (eds.) KR 1996: Principles of Knowledge Representation and Reasoning, pp. 186–197. Morgan Kaufmann, San Francisco (1996)
332. Fikes, R., Nilsson, N.: STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence 2, 189–208 (1971)
333. Blum, A., Furst, M.: Fast planning through planning graph analysis. Artificial Intelligence 90, 281–300 (1997)
334. Ernst, M., Millstein, T.D., Weld, D.S.: Automatic SAT-compilation of planning problems. In: Proceedings of the IJCAI 1997, pp. 1169–1177 (1997)
335. Kautz, H., Selman, B.: Unifying SAT-based and graph-based planning. In: Minker, J. (ed.) Proceedings of the Workshop on Logic-Based Artificial Intelligence, Washington, DC, College Park, Maryland, Computer Science Department, University of Maryland, June 14–16 (1999)
336. Kautz, H.A., McAllester, D., Selman, B.: Encoding plans in propositional logic. In: Proceedings of the Fifth International Conference on the Principle of Knowledge Representation and Reasoning (KR 1996), pp. 374–384 (1996)
337. Kautz, H.A., Selman, B.: Planning as satisfiability. In: Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI 1992), pp. 359–363 (1992)
338. Gerevini, A., Serina, I.: LPG: a planner based on planning graphs with action costs. In: Proceedings of the Sixth International Conference on AI Planning and Scheduling, pp. 13–22. AAAI Press, Menlo Park (2002)
339. Bylander, T.: A linear programming heuristic for optimal planning. In: Proceedings of the 14th National Conference on Artificial Intelligence (AAAI 1997), Providence, Rhode Island, pp. 694–699. AAAI Press/MIT Press, Menlo Park (1997)
340. Gałuszka, A., Świerniak, A.: Translation STRIPS planning in multi-robot environment to linear programming. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) ICAISC 2004. LNCS, vol. 3070, pp. 768–773. Springer, Heidelberg (2004)
341. van Beek, P., Chen, X.: Cplan: A constraint programming approach to planning. In: Proceedings of the 16th National Conference on Artificial Intelligence (IJCAI 1999), pp. 585–590 (1999)
342. Veloso, M.: Planning and Learning by Analogical Reasoning. Springer, Heidelberg (1994)
343. Vere, S.: Planning in time: Windows and durations for activities and goals. IEEE Transactions on Pattern Analysis and Machine Intelligence 5, 246–267 (1983)

344. Sacerdoti, E.: The nonlinear nature of plans. In: Proceedings of the Fourth International Joint Conference on Artificial Intelligence (IJCAI 1975), pp. 206–214 (1975)
345. Tate, A.: Generating project networks. In: Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI 1977), pp. 888–893 (1977)
346. TLPlan: Project web site, `http://www.cs.toronto.edu/~fbacchus/tlplan.html`
347. Doherty, P., Kvarnstrom, J.: TALplanner: A temporal logic-based planner. AI Magazine 22, 95–102 (2001)
348. TALplanner: Project web site, `http://www.ida.liu.se/~patdo/aiicssite1/kplab/projects/talplanner/`
349. Anzai, Y.: Pattern recognition and machine learning. Academic Press, San Diego (1992)
350. Bernardinello, L., Cindio, F.D.: A survey of basic net models and modular net classes. In: Rozenberg, G. (ed.) APN 1992. LNCS, vol. 609. Springer, Heidelberg (1992)
351. Suyama, T., Yokoo, M.: Strategy/false-name proof protocols for combinatorial multi-attribute procurement auction. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), New York, NY, USA, August 19-23, 2004, pp. 160–167. IEEE Computer Society, Los Alamitos (2005)
352. Yokoo, M.: Protocol/mechanism design for cooperation/competition. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), New York, NY, USA, August 19-23, pp. 3–7. IEEE Computer Society, Los Alamitos (2005)
353. Góra, G., Bazan, J.G., Kruczek, P., Bazan-Socha, S., Skowron, A., Pietrzyk, J.: Case-based planning of treatment of infants with respiratory failure. In: Czaja, L. (ed.) Proceedings Workshop on Concurrency, Specification, and Programming (CS&P 2007), Łagów, Poland, Warsaw, Warsaw University, September 27-28, pp. 223–234 (2007)
354. Góra, G., Bazan, J.G., Kruczek, P., Bazan-Socha, S., Skowron, A., Pietrzyk, J.: Case-based planning of treatment of infants with respiratory failure. Fundamenta Informaticae (to appear, 2008)
355. Simulator, R.: Project web site, `http://logic.mimuw.edu.pl/~bazan/simulator`

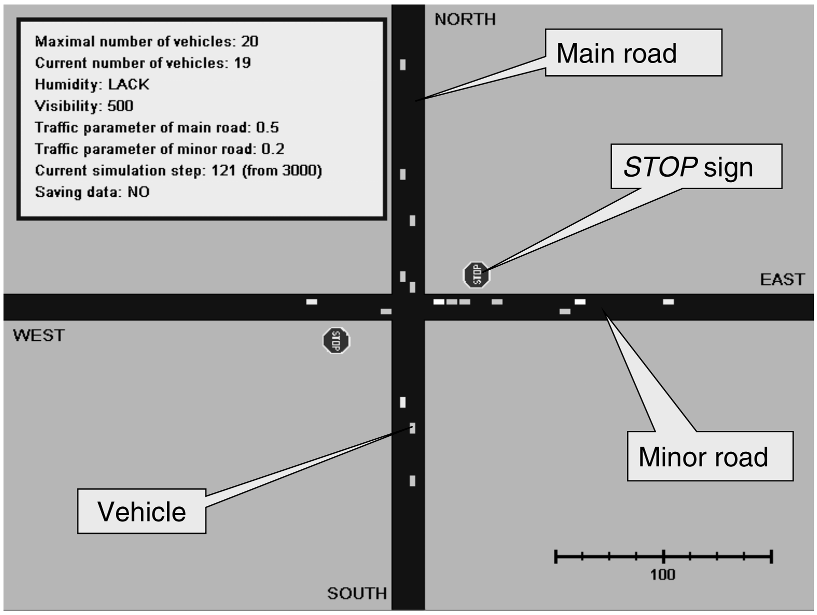
A Road Simulator

Road simulator is a tool for generating data sets recording vehicle movement on the road and at the crossroads (see [179, 355]). Such data is extremely crucial in testing complex decision systems monitoring the situation on the road that are working on the basis of information coming from different devices. The simulator was constructed by the author of this paper.

Driving simulation takes place on a board (see Fig. 54) which presents a crossroad together with the access roads.

During the simulation the vehicles may enter the board from all four directions, that is, East, West, North, and South. The vehicles coming to the crossroad from South and North have the right of way in relation to the vehicles coming from West and East.

Each of the vehicles entering the board has only one aim, i.e., to drive through the crossroad safely and to leave the board. The simulation takes place step by



Maximal number of vehicles: 20
Current number of vehicles: 19
Humidity: LACK
Visibility: 500
Traffic parameter of main road: 0.5
Traffic parameter of minor road: 0.2
Current simulation step: 121 (from 3000)
Saving data: NO

Fig. 54. The board of simulation

step and at each of its steps the vehicles may perform the following maneuvers during the simulation: passing, overtaking, changing direction (at the crossroad), changing lane, entering the traffic from the minor road into the main road, stopping, and pulling out.

Planning further steps of each vehicle takes place independently at every step of the simulation. Each vehicle is "observing" the surrounding situation on the road, keeping in mind its destination and its own parameters (driver's profile), makes an independent decision about its further steps; whether it should accelerate, decelerate and what (if any) maneuver should be commenced, continued, ended or stopped.

Making decisions concerning further driving, a given vehicle takes under consideration its parameters and the driving parameters of five vehicles next to it which are marked by FR1, FR2, FL, BR, and BL (see Fig. 55).

During the simulation the system registers a series of parameters of the local simulations, that is, simulations connected with each vehicle separately, as well as two global parameters of the simulation, that is, parameters connected with driving conditions during the simulation. The value of each simulation parameter may vary and can be treated as a certain attribute taking values in a specified value set.

We associate the simulation parameters with the readouts of different measuring devices or technical equipment placed inside the vehicle or in the outside environment (*e.g.*, by the road, in a helicopter observing the situation on the road, in a police car). These are devices and equipment playing the role of detecting devices

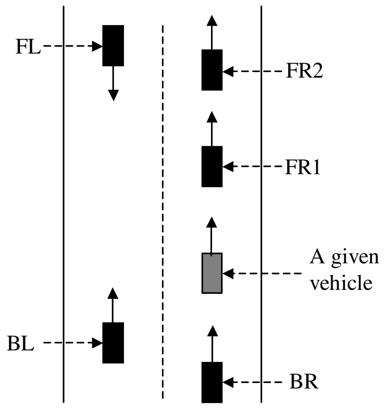


Fig. 55. A given vehicle and five vehicles next to it

or converters meaning sensors (*e.g.*, a thermometer, range finder, video camera, radar, image, and sound converter). The attributes taking the simulation parameter values, by analogy to devices providing their values, are called sensors.

The exemplary sensors are the following: initial and current road (four roads), distance from the crossroad (in screen units), current lane (two lanes), position of the vehicle on the road (values from 0.0 to 1.0), vehicle speed (values from 0.0 to 10.0), acceleration and deceleration, distance of a given vehicle from the vehicles FR1, FL, BR, and BL and between FR1 and FR2 (in screen units), appearance of the vehicle at the crossroad (binary values), visibility (expressed in screen units, values from 50 to 500), humidity (slipperiness) of the road (three values: lack of humidity (dry road), low humidity, and high humidity).

If, for some reason, the value of one of the sensors is not determined, the value of the parameter will become equal to NULL (missing value).

Apart from sensors, the simulator registers a few more attributes whose values are determined using the sensor values in a way determined by an expert. These parameters take the binary values and are therefore called concepts in the present simulator version. The results returned by testing concepts are very often in the form YES, NO or DOES NOT CONCERN (NULL value).

Here are exemplary concepts:

1. Is a vehicle forcing the right of way at the crossroad?
2. Is there free space on the right lane in order to end the overtaking maneuver?
3. Will a vehicle be able to overtake easily before the oncoming car?
4. Will a vehicle be able to brake before the crossroad?
5. Is the distance from the vehicle FR1 too short or do we predict that it may happen shortly?
6. Is a vehicle overtaking safely?
7. Is a vehicle driving safely?

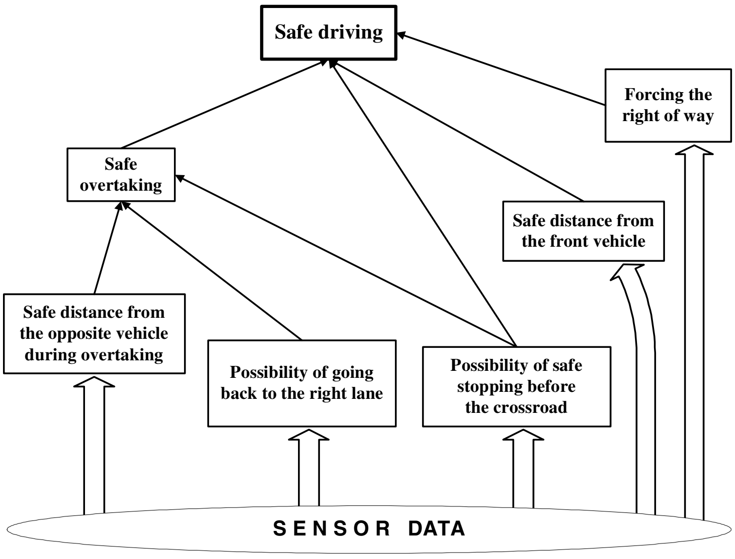

Safe driving
Forcing the right of way
Safe overtaking
Safe distance from the front vehicle
Safe distance from the opposite vehicle during overtaking
Possibility of going back to the right lane
Possibility of safe stopping before the crossroad
SENSOR DATA

Fig. 56. The relationship diagram for the presented concepts

Besides binary concepts, simulator registers for any such concept one special attribute that approximates binary concept by six linearly ordered layers: *certainly YES*, *rather YES*, *possibly YES*, *possibly NO*, *rather NO*, and *certainly NO*.

Some concepts related to the situation of the road are simple and classifiers for them can be induced directly from sensor measurement but for more complex concepts this is infeasible. In searching for classifiers for such concepts domain knowledge can be helpful. The relationships among concepts represented in a domain knowledge can be used to construct hierarchical relationship diagrams. Such diagrams can be used to induce multi-layered classifiers for complex concepts (see, *e.g.*, [52, 182]). In Fig. 56 there is an exemplary relationship diagram for the concepts mentioned above .

The concept specification and concept dependencies are usually not given automatically in accumulated data sets. Therefore, they should be extracted from a domain knowledge. Hence, the role of human experts is very important in our approach.

During the simulation, when a new vehicle appears on the board, its so-called driver's profile is determined. It may take one of the following values: a very careful driver, a careful driver, and a careless driver. Driver's profile is the identity of the driver and according to this identity further decisions as to the way of driving are made.

Driver's profile is determined at the beginning when the vehicle appears on the board and cannot be changed until it disappears from the board.

For a given vehicle the driver's profile is determined randomly by the following probability distribution: a very careful driver 0.4, careful driver 0.25, and careless driver 0.35.

Depending on the driver's profile and weather conditions (humidity of the road and visibility), speed limits are determined which cannot be exceeded.

The humidity of the road influences the length of braking distance for depending on humidity, different speed changes take place within one simulation step, with the same braking mode.

The driver's profile influences the speed limits dictated by visibility. If another vehicle is invisible for a given vehicle, this vehicle is not taken into consideration in the independent planning of further driving by a given car. Because this may cause dangerous situations, depending on the driver's profile, there are speed limits for the vehicle.

The data generated during the simulation are stored in a data table (information system). Each row of the table depicts the situation of a single vehicle and the sensor and concept values are registered for a given vehicle and the vehicles FR1, FR2, FL, BL, and BR (associated with a given vehicle). Within each simulation step, descriptions of situations of all the vehicles on the road are saved to a file.

B Neonatal Respiratory Failure

The new possibilities in medical intensive care have appeared during last decades thanks to the progress in medical and technical sciences. This progress allowed us to save the live of prematurely born infants including the smallest born between the 20th and the 24th week of gestation with the birth weight above 500g.

Prematurely born infants demonstrate numerous abnormalities in their first weeks of life. Their survival, especially without severe multiorgan complications is possible with appropriate treatment. Prematurity can be characterized as inappropriate maturity of systems and organs leading to their dysfunction after birth.

The respiratory system dysfunction appearing in the first hours of life and leading to respiratory failure is the most important single factor limiting survival of our smallest patients. The respiratory failure is defined as inappropriate blood oxygenation and accumulation of carbon dioxide and is diagnosed based on arterial blood gases measurements. Clinical symptoms increased rate of breathing, accessory respiratory muscles use as well as X-ray lung examination are also included in assessment of the severity of respiratory failure (see, e.g, [328] for more details).

The most important cause of respiratory failure in prematurely born infants is RDS (*respiratory distress syndrome*). RDS is evoked by lung immaturity and surfactant deficiency. The other co-existing abnormalities PDA[5] (*patent duc-*

[5] PDA (patent ductus arteriosus) is a heart problem that occurs soon after birth in some infants. In PDA, there is an abnormal circulation of blood between two of the major arteries near the heart. Before birth, the two major arteries, i.e., the aorta and the pulmonary artery, are normally connected by a blood vessel called the *ductus arteriosus*, which is an essential part of the fetal circulation. After birth, the vessel is supposed to close within a few days as part of the normal changes occurring in the

tus arteriosus), sepsis (generalized reaction on infection leading to multiorgan failure), and Ureaplasma lung infection (acquired during pregnancy or birth) may exacerbate the course of respiratory failure. Each of these conditions can be treated as an unrelated disease requiring a separate treatment. But they co-exist in a patient very often, so in a single patient we may deal with their combination, for example, RDS + PDA + sepsis. In the holistic therapeutic approach, it is important to synchronize the treatment of the co-existing abnormalities, which can finally lead to cure from the respiratory failure.

The respiratory failure dominates in clinical course of prematurity but is not the only factor limiting the success of treatment. Effective care of the prematurely born infant should include all co-existing abnormalities such as infections, both congenital and acquired, water-electrolyte and acid-base imbalance, circulatory, kidney, and other problems. All these factors are related and they influence one another. The care of the prematurely born infants in their first days of life requires continuous analysis of plenty of the parameters including vital sings and the results of the additional tests. These parameters can be divided into stationary (*e.g.*, gestational age, birth weight, Apgar score) and continuous changing in time. The continuous values can be examined on the discrete (*e.g.*, blood gases) or the continuous basis, *e.g.*, with the monitoring devices (oxygen hemoglobin saturation, hear rate, blood pressure, temperature, and lung mechanics). The neonatal care includes assessment of imagine techniques results (ultrasound of the brain, echocardiography, chest X-ray). The global analysis should also include current methods of treatment applied in the particular patients. They may have qualitative (*e.g.*, administration of medication) or quantitative (*e.g.*, respiratory settings) characteristics.

Everyday analysis of numerous parameters requires great theoretical knowledge and practical experience. It is worth mentioning that this analysis should be quick and precise. Assessment of the patient's state is performed very often under rush and stress conditions.

A very important element of this analysis is an appropriate assessment of the risk of death of the small patient caused by the respiratory failure during next hours or days. The appropriate assessment of this risk leads to the decision of a particular method and level of treatment. The life of a sick child depends on this quick and correct decision. It should be emphasized that the correct assessment of the risk of death depends not only on analysis of the current clinical status, lab tests, and imagine techniques results but also on the dynamics observed lately and the character of changes (*e.g.*, progression of the blood gases indices of respiratory failure). The additional risk parameters such as birth weight are also important.

Computer techniques can be very useful in the face of difficulties in an effective data analysis. They may provide a support for the physician in everyday

infant's circulation. In some infants, however, the ductus arteriosus remains open (patent). If an infant has a PDA, but has an otherwise normal heart, the PDA may shrink and go away completely. If a PDA does not shrink, or is due to causes other than prematurity, surgery may be needed. This surgery is called *ligation* and involves placing a suture around the ductus to close it (see, e.g, [328] for more details).

diagnostic-therapeutic process both as a collecting, storing and patient's data presenting tools (*e.g.*, Neonatal Information System [329]) and as a tool of quick, automatic and intelligent analysis of this data. This approach might enable a computer presentation of some information based on the observed patterns which might be helpful in planning of the treatment. An example is the tool detecting patterns of changes in the newborn clinical status which lead to death with high probability. This kind of patterns is called the risk patterns (see Section 6.23). In this approach, a given patient is treated as an investigated complex dynamical system, whilst diseases of this patient (RDS, PDA, sepsis, Ureaplasma, and the respiratory failure) are treated as complex objects changing and interacting over time (see Section 6.22).

Index of Main Symbols

Symbol	Description	Page
CDS	A complex dynamic system	478
RSES	The Rough Set Exploration System	502
$\mathbf{A} = (U, A)$	An information system	503
V_a	A value set of an attribute a	503
$\mathbf{A} = (U, A, d)$	A decision table	503
$IND_{\mathbf{A}}(B)$	A B-indiscernibility relation in a decision table $\mathbf{A}$	504
$[u]_{IND_{\mathbf{A}}(B)}$	An equivalence class of a relation $IND_{\mathbf{A}}(B)$, such that an object u belongs to this class	504
$RED_{\mathbf{A}}(A)$	A set of all reducts in an information system $\mathbf{A}$	504
$M(\mathbf{A})$	A discernibility matrix of an information system $\mathbf{A}$	504
$f_{\mathbf{A}}$	A discernibility function for an information system $\mathbf{A}$	504
$\underline{B}X$, $\overline{B}X$	B-lower and B-upper approximations of a set X in an information system $\mathbf{A}$	505
$BN_B(X)$	B-boundary of a set X (boundary region)	505
$POS_B(d)$	A B-positive region of a decision table $\mathbf{A} = (U, A, d)$	505
$RED(\mathbf{A}, d)$	A set of all relative reducts of a decision table $\mathbf{A}$	506
$Pred(r)$	A predecessor of a rule r	508
$Succ(r)$	A successor of a rule r	508
$Match_{\mathbf{A}}(r)$	A number of objects matched by a decision rule r	508
$Supp_{\mathbf{A}}(r)$	A number of objects supporting a decision rule r	508
$\mu_{\mathbf{A}}(r)$	A coefficient of consistency of a rule r	508

$RUL(\mathbf{A})$	A set of all optimal basic decision rules of an information system $\mathbf{A}$, i.e., a set of all rules with minimal numbers of descriptors of an information system $\mathbf{A}$	508
$\mathbf{A}(T)$	A sub-table of a decision table $\mathbf{A}$ containing all objects matching a template T	512
$\mathbf{A}(\neg T)$	A sub-table of a decision table $\mathbf{A}$ containing all objects not matching a template T	512
μ_C	A rough membership function for a concept C	514
$AS = (U, I, \nu)$	An approximation space	515
$LOW\,(AS, X)$	A lower approximation of a set X with respect to an approximation space AS	516
$UPP\,(AS, X)$	An upper approximation of a set X with respect to an approximation space AS	516
μ_C^E	A stratifying classifier of a concept C which classifies each tested object into one of the layers labeled with labels from a set E	526
$\mathcal{L}$	A language to define features of complex objects (a set of formulas over a given finite alphabet)	547
$\models_{\mathcal{L}}$	A satisfiability relation of a language $\mathcal{L}$	547
$\mathcal{P} = (X, \mathcal{L}, \models)$	A property system	547
$GDL(\mathbf{A})$	A generalized descriptor language of an information system $\mathbf{A}$	549
$DISM_{\mathbf{A}}$	A dissimilarity function of pairs of objects in an information system $\mathbf{A}$	550
$NL(\mathbf{A})$	A neighborhood language for an information system $\mathbf{A}$	551
$\mu_{DISM_{\mathbf{A}}}$	A dissimilarity classifier for an information system $\mathbf{A}$	552
TIS	A total information system	553
$PEC(\mu_C^E)$	A language of patterns extracted from a classifier μ_C^E	575
$\mathbf{A}_C$	A concept approximation table using stratifying classifiers for a concept C	577
$\mathbf{LT}_C$	A layer table for a concept C	581
AR-scheme	An approximate reasoning scheme	588
$\mathbf{T}$	A temporal information system	596
$ETW(\mathbf{T})$	A language for extracting time windows from system $\mathbf{T}$	600
$TW(\mathbf{T})$	A family of all time windows from a c-temporal information system $\mathbf{T}$	601
$TW(\mathbf{T}, s)$	A family of all time windows from a c-temporal information system $\mathbf{T}$ with length equals to s	601

$FTW(\mathbf{T})$	A language for definnig features of time windows of c-temporal information system $\mathbf{T}$	603
$\overline{\mathbf{T}} = (\overline{U}, \overline{A})$	An information system of time windows (TW-information system)	605
$ECTW(\mathbf{T})$	A language for extracting clusters of time windows from a system $\mathbf{T}$ ($ECTW$-language)	610
$FCTW(\overline{\mathbf{T}})$	A information system of time windows for a system $\mathbf{T}$ ($FCTW$-language)	612
$\overline{\overline{\mathbf{T}}} = (\overline{\overline{U}}, \overline{\overline{A}})$	An information system of clusters of time windows (CTW-information system)	613
$\overline{\overline{\mathbf{T}}}_C = (\overline{\overline{U}}, \overline{\overline{A}} \cup \{d_C\})$	A temporal concept table for a concept C	614
$G = (V, E)$	A behavioral graph	617
$PATH(G, l)$	A family of all temporal paths with a length l in a behavioral graph G	617
SA	A sweeping algorithm around a complex object	620
$ESTW(\mathbf{T})$	A language for extracting sequences of time windows from system $\mathbf{T}$ ($ESTW$-language)	625
$SEQ(\mathbf{T})$	A family of all sequences of time windows of a given c-temporal information system $\mathbf{T}$	626
$SEQ(\mathbf{T}, s)$	A family of all sequences of time windows of a length s of a given c-temporal information system $\mathbf{T}$	626
$SEQ(\mathbf{T}, l, s)$	A family of all sequences of time windows of a length s of a given c-temporal information system $\mathbf{T}$ that they are windows of a length l	626
$Subsequence(S, i, j)$	A sub-sequence of a sequence of time windows S, where i and j are indexes of time windows, such that $i < j$	626
$Partition(W, k)$	A k-partition of a time window W	627
$FTP(\mathbf{G})$	A language for defining features of temporal paths of behavioral graph $\mathbf{G}$	629
$\overline{\mathbf{G}} = (\overline{U}, \overline{A})$	An information system of temporal paths (TP-information system)	630
$DPATH(\mathbf{G}, s)$	A set of all temporal paths observed in data of duration s	630
$ECTP(\mathbf{G})$	A language for extracting clusters of temporal paths from a behavioral graph $\mathbf{G}$ ($ECTP$-language)	633
$FCTP(\mathbf{G})$	A language for definnig features of clusters of temporal paths for a behavioral graph $\mathbf{G}$ ($FCTP$-language)	636
$\overline{\overline{\mathbf{G}}} = (\overline{\overline{U}}, \overline{\overline{A}})$	An information system of clusters of temporal paths (CTP-information system)	637

$\mathbf{G}_R = (V_R, E_R)$	A behavioral graph representing changes of relations between parts of structured objects	640
$\mathbf{G}_{\bowtie} = (U_{\bowtie}, A_{\bowtie})$	An information system of clusters of temporal paths for structured objects ($SCTP$-information system)	640
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	A behavioral graph for a structured object	644
$PATH(\mathcal{G}, l)$	A family of all temporal paths with length l in a behavioral graph $\mathcal{G}$	644
$\mathbf{PG} = (S, A, E)$	A planning graph	663
$PATH(\mathbf{PG})$	A family of all paths in a planning graph $\mathbf{PG}$	663
$PATH(\mathbf{PG}, k)$	A family of all paths with length k in a planning graph $\mathbf{PG}$	663
$Subpath(p, i, j)$	A sub-path of a path p, where i and j are indexes of nodes, such that $i < j$	663
$PLAN(\mathbf{PG})$	A family of all plans in a planning graph $\mathbf{PG}$	663
$PLAN(\mathbf{PG}, k)$	A family of all plans with length k in a planning graph $\mathbf{PG}$	663
$Subplan(p, i, j)$	A sub-plan of a plan p, where i and j are indexes of states, such that $i < j$	663
$FPPG(\mathbf{PG})$	A language for defining features of paths in planning graphs ($FPPG$-language)	668
$TW(\mathbf{T}, p)$	A set of all time windows of a given plan p in system $\mathbf{T}$	673
$DPLAN(\mathbf{T}, \mathbf{PG}, k)$	A set of all plans, occurring in system $\mathbf{T}$ of a length k	673
$DPLAN(\mathbf{T}, \mathbf{PG}, s, k)$	A set of all plans, occurring in system $\mathbf{T}$ and ending with state s and of a length k	673
$\mathbf{RT}(s, k)$	A resolving table for a state s from a planning graph constructed using a length of path k	673
MAP1	The low-intensity mechanical ventilation	675
MAP2	The middling-intensity mechanical ventilation	675
MAP3	The high-intensity mechanical ventilation	675
CPAP	The continuous positive airway pressure	675
$\mu_{\mathbf{RT}(s,k)}$	A resolving classifier constructed for a table $\mathbf{RT}(s, k)$	676
$PairList(\mu_{\mathbf{RC}(s,k)}(p))$	A list of pairs $(action, state)$ which a classifier $\mu_{\mathbf{RC}(s,k)}$ returns to a path p	676
EFS	The expert forward search algorithm	678
$EEFS$	The exhaustive expert forward search algorithm	679
$FEEFS$	The full exhaustive expert forward search algorithm	680
$DISM_{\mathbf{PG}}$	A function of dissimilarity between states from a planning graph $\mathbf{PG}$	683

$\mathbf{DIT}_{\mathbf{PG}}$	A table of dissimilarity between states from a planning graph $\mathbf{PG}$	683
$\mu_{DIT(\mathbf{PG})}$	A classifier of dissimilarity between states from a planning graph $\mathbf{PG}$	683
Reconstruction	The algorithm of partial reconstruction of a plan for unstructured objects	686
$\mathcal{PG} = (\mathcal{S}, \mathcal{A}, \mathcal{E})$	A planning graph for structured objects	688
$\mathbf{PGF}$	A family of planning graphs	690
$GPLAN(\mathbf{PGF}, k)$	A set of all g-plans with length k for a family of planing graphs $\mathbf{PGF}$	690
$DGPLAN(\mathbf{T}, \mathbf{PGF}, k)$	A set of all g-plans occurring in a system $\mathbf{T}$ with length k and constructed for a family of planing graphs $\mathbf{PGF}$	690
$\mathbf{ET}(\mathbf{T}, \mathbf{PGF}, k)$	An elimination table of g-plans from a set $GPLAN(\mathbf{PGF}, k)$	691
$\mathbf{ERUL}(\mathbf{T}, \mathbf{PGF}, k)$	A set of all elimination rules for an elimination table $\mathbf{ET}(\mathbf{T}, \mathbf{PGF}, k)$	693
$\mu_{\mathbf{ET}(\mathbf{T}, \mathbf{PGF}, k)}$	An elimination classifier based on a set $\mathbf{ERUL}(\mathbf{T}, \mathbf{PGF}, k)$	694
$\mathbf{MAT}(\mathbf{T}, \mathbf{PGF}, k)$	A meta action table of g-plans for structured objects	695
$\mu_{\mathbf{MAT}(\mathbf{T}, \mathbf{PGF}, k)}$	A meta action classifier constructed for a table $\mathbf{MAT}(\mathbf{T}, \mathbf{PGF}, k)$	695
SearchGPlanMA	The algorithm of searching of g-plan for a meta action	699
SearchGPlan	The algorithm of searching of g-plan for states	700
EEFSS	The exhaustive expert forward search algorithm for a structured object	701
GReconstruction	The algorithm of partially reconstruction of a g-plan	706
MReconstruction	The algorithm of reconstruction of a plan for structured objects	707
RDS	The respiratory distress syndrome	744
PDA	Patent ductus arteriosus	744

Author Index