Chapter 12

Rough Sets and Rough Logic: A KDD Perspective

Zdzisław Pawlak[1], *Lech Polkowski*[2], *and Andrzej Skowron*[3]

[1] Institute of Theoretical and Applied Informatics
Polish Academy of Sciences
Bałtycka 5 44000 Gliwice Poland
[2] Polish-Japanese Institute of Information Technology
Koszykowa 86 02008 Warszawa Poland
Department of Mathematics and Information Sciences
Warsaw University of Technology
Pl. Politechniki 1 00665 Warszawa Poland

[3] Institute of Mathematics Warsaw University
Banacha 2 02097 Warszawa Poland
emails: zpw@ii.pw.edu.pl; polkow@pjwstk.waw.pl; skowron@mimuw.edu.pl

Abstract Basic ideas of rough set theory were proposed by Zdzisław Pawlak [85, 86] in the early 1980's. In the ensuing years, we have witnessed a systematic, world–wide growth of interest in rough sets and their applications.

The main goal of rough set analysis is induction of approximations of concepts. This main goal is motivated by the basic fact, constituting also the main problem of KDD, that languages we may choose for knowledge description are incomplete. A fortiori, we have to describe concepts of interest (features, properties, relations etc.) not completely but by means of their reflections (i.e. approximations) in the chosen language. The most important issues in this induction process are:

- construction of relevant primitive concepts from which approximations of more complex concepts are assembled,
- measures of inclusion and similarity (closeness) on concepts,
- construction of operations producing complex concepts from the primitive ones.

Basic tools of rough set approach are related to concept approximations. They are defined by approximation spaces. For many applications, in particular for KDD problems, it is necessary to search for relevant approximation spaces in the large space of parameterized approximation spaces. Strategies for tuning parameters of approximation spaces are crucial for inducing concept approximations of high quality.

Methods proposed in rough set approach are kin to general methods used to solve Knowledge Discovery and Data Mining (KDD) problems like feature selection, feature extraction (e.g. discretization or grouping of symbolic value),

data reduction, decision rule generation, pattern extraction(templates, association rules), or decomposition of large data tables. In this Chapter we examine rough set contributions to Knowledge Discovery from the perspective of KDD as a whole.

This Chapter shows how several aspects of the above problems are solved by the classical rough set approach and how they are approached by some recent extensions to the classical theory of rough sets. We point out the role of Boolean reasoning in solving discussed problems. Rough sets induce via its methods a specific logic, which we call rough logic. In the second part of this Chapter, we discuss rough logic and related incomplete logics from a wider perspective of logical approach in KDD.

Keywords: indiscernibility, lower and upper approximations, rough sets, boundary region, positive region, rough membership function, decision rules, patterns, dependencies in a degree, rough mereology, logic, propositional logic, predicate logic, modal logic, belief and belief revision logics, default logic, para–consistent logic, non-monotonic logic, fuzzy logic, rough logic.

1 Basic philosophy of rough set methods in KDD

Rough set tools for KDD are based on parameterized approximation spaces and strategies for parameter tuning leading to approximations of (whole or parts of) sets of objects called *decision classes* (or *concepts*).

Approximation spaces (cf. Chapter by STEPANIUK) consist of two main constructs: sets of objects called *neighborhoods* and a measure of inclusion on sets. Neighborhoods are described by some parameterized formulas. The inclusion measure is indexed by some parameters describing a degree of set inclusion. By parameter tuning the relevant neighborhoods as well as the relevant degree of inclusion are chosen with respect to a given problem. In this section we explain the main steps in inducing concept (or its part) description on the basis of rough set methods.

We assume a finite universe U of objects, represented in an information system $\mathcal{A} = (U, A)$ (see Chapter 1), being a subset of the whole universe U^∞ of objects. By a *concept* (*decision class*) in U^∞ we understand any subset of U^∞.

One of the main goals of such areas as Machine Learning, Pattern Recognition as well as Knowledge Discovery and Data Mining is to develop methods for approximate descriptions of concepts (or their interesting parts) on the basis of partial information about them represented by information about objects from the universe U. More formally, let $X \subseteq U^\infty$ be a concept and let $\mathcal{A} = (U, A)$ be a given information system. (The information system together with the characteristic function of $X \cap U$ are represented by so called *decision systems*.) The problem is to induce an approximate description of $U \cap X$ using attributes from A so that its extension to U^∞ will approximate X with sufficiently high quality. One could formulate this by saying that the induced concept description should estimate the real concept well.

Let us mention that there are some other factors due to which a concept approximation can be induced rather than its exact description. Among them are well known problems with missing values or noise in data.

Rough set approach allows to precisely define the notion of a concept approximation. It is based [86] on the *indiscernibility* (or, a weaker form viz. a similarity) relation between objects defining a *partition* (or, covering) of the universe U of objects. There are some issues to be discussed here.

The first one concerns the way in which inclusion of sets, in particular, of neighborhoods is measured.

From the point of view of KDD, it is important to consider two notions related to formulas, namely *inclusion* and *closeness*. Instead of classical crisp inclusion or equivalence it is more appropriate to consider inclusion in a degree and closeness in a degree. An example of such inclusion is used in case of association rules [4]. Assuming a given set of formulas and a given finite universe of objects the task is to search for pairs of formulas α, β such that α is sufficiently included in β. The degree of inclusion for association rules is expressed by two thresholds consisting of coefficients known as *support* and *confidence*.

The second issue concerns the definition of quality measure used to estimate quality of concept description by neighborhoods from a given partition (or covering) in U and in U^{∞}.

To solve the first problem, i.e., approximation in U, rough sets are offering the approximation operations on sets of objects (see Chapter 1). The classical rough set approach [86] is using crisp inclusion and by taking the union of all neighborhoods included in a given set it yields the *lower approximation* of this set. One can also define the *upper approximation* of the given set of objects by taking union of all objects having non-empty intersection with a given set. In the complement of the upper approximation of a given set there are all objects which can be with certainty classified as not belonging to the set (using available neighborhoods). The difference between the upper and lower approximations is called the *boundary region* of the approximated set and it is created by objects which cannot be classified with certainty neither to the approximated set nor to its complement (using given neighborhoods defining partition or covering). Consequently, each rough set exhibits *boundary-line cases*, i.e., objects which cannot be with certainty classified neither as members of the set nor of its complement.

As a consequence vague concepts, in contrast to precise concepts, cannot be characterized in terms of information about their elements. Therefore, in the proposed approach, we assume that any vague concept is replaced by a pair of precise concepts – the lower and the upper approximation of this vague concept. The difference between the upper and the lower approximation constitutes the boundary region of the vague concept.

For applications in KDD, a more relaxed approach has been developed: instead of crisp inclusion it seems more appropriate to use inclusion in a degree and to define set approximations on the basis of such inclusion [137], [112], [92], [88], [113]. One can search for proper degrees in the sense that induced by their means concept descriptions are showing better classification quality on new objects or,

more generally, they give better quality of concept description in U^{∞}.

There are some other aspects of concept description which should be taken into consideration when one would like to estimate its quality in U^{∞}. One can follow, e.g., the *minimal description length principle* [103] (cf. Chapter by ŚLĘZAK) suggesting that more coarser partitions of U allow for better concept approximation assuming they describe the set $X \cap U$ still with a sufficient quality (measured e.g. by the size of boundary region or other measures like entropy). Hence, in particular, it follows that preprocessing methods, leading to partitions more coarser yet still relevant for concept approximation, are important for rough set approach. Moreover, it is necessary to develop methods for achieving an appropriate tradeoff between coarseness of partitions and quality of concept approximation determined by the chosen partition (or covering). There are several approaches, known in Machine Learning or Pattern Recognition realizing this tradeoff. Among them one can distinguish the following ones:

1. **Feature selection** is realized by searching for relevant indiscernibility relations in the family of all indiscernibility relations $IND(B)$ where $B \subseteq A$ (see Chapter 1).
2. **Feature extraction** is realized by searching for relevant indiscernibility relations in the family of all indiscernibility relations $IND(B)$ where $B \subseteq C$ and C is a given set of attributes in A (see Chapter 1).
3. **Discovery of a relevant subspace C of attributes definable by A.** Next, the relevant attributes are extracted within a given subspace C of attributes definable by A. In this way the searching process for relevant attributes can be made more efficient because the computational complexity of the searching problem for relevant attributes in the whole space of attributes definable by A makes the process of relevant feature extraction infeasible.

It is worth to mention that very little is known about how to discover relevant subspaces C of attributes definable by A and this is still a challenge for knowledge discovery and data mining.

The (described above) process of searching for appropriate (coarser yet relevant) partitions (defined by C) is related in rough set theory to searching for constructs called *reducts* and their approximations. Let us observe that in practice the generated models of partitions (coverings) are usually sub-optimal with respect to the minimal description length principle. This is because of the computational complexity of searching problems for the optimal models. Moreover, it is necessary to tune the extracted models to receive satisfactory quality, e.g. of classification or description of induced models.

The combination of rough set approach with Boolean Reasoning [13] has created a powerful methodology allowing to formulate and efficiently solve searching problems for different kinds of reducts and approximations to them.

The idea of Boolean reasoning consists in constructing – for a given problem P – a corresponding Boolean function f_P with the following property: solutions for the problem P can be recovered from prime implicants [4] of the Boolean function f_P.

[4] An implicant of a Boolean function f is any conjunction of literals (variables or

It is necessary to deal with Boolean functions of large size to solve real – life problems. A successful methodology based on the discernibility of objects and Boolean reasoning (see Chapter 1) has been developed for computing many important for applications constructs like reducts and their approximations, decision rules, association rules, discretization of real value attributes, symbolic value grouping, searching for new features defined by oblique hyperplanes or higher order surfaces, pattern extraction from data as well as conflict resolution or negotiation. Reducts are also basic tools for extracting functional dependencies or functional dependencies in a degree from data(for references see the papers and bibliography in [110], [82], [94], [95]).

A great majority of problems related to generation of the above mentioned constructs are of high computational complexity (i.e. they are NP-complete or NP-hard). This is also showing that most problems related to e.g. feature selection or pattern extraction from data have intrinsic high computational complexity. However, using the methodology based on discernibility and Boolean reasoning it has been possible to discover efficient heuristics returning suboptimal solutions to these problems. The reported results of experiments on many data sets are very promising. They show very good quality of solutions (expressed by the classification quality of unseen objects and time necessary for solution construction) generated by proposed heuristics in comparison to other methods reported in literature. Moreover, for large data sets the decomposition methods based on patterns called templates have been developed (see e.g. [79], [76]) as well as a method to deal with large relational databases (see e.g. [75]).

It is important to note that this methodology allows for constructing heuristics having a very important *approximation property* which can be formulated as follows: expressions generated by means of heuristics (i.e., implicants) and sufficiently *close* to prime implicants define approximate solutions to the problem [108].

Parameterized approximation spaces and strategies for parameter tuning are basic tools for rough set approach in searching for data models under a given partition of objects (see e.g. [79], [76],[110], [116], [117], [94], [95]). Parameters to be tuned are thresholds used to select elements of partitions (coverings), to measure degree of inclusion (or closeness) of sets, or parameters measuring quality of approximation.

Rough sets offer methods for exploratory data analysis, i.e., methods for hypothesis generation rather than hypothesis testing [39]. Data mining without proper consideration of statistical nature of the inference problem is indeed to be avoided. We now shortly discuss how statistical methods are used in combination with rough set methods.

There is an important issue about the statistical validity and significance of constructs generated using rough set methods. Rough set data analysis becomes

---

their negations) such that if the values of these literals are true under an arbitrary valuation v of variables then the value of the function f under v is also true. A prime implicant is a minimal implicant. Here we are interested in implicants of monotone Boolean functions only, i.e. functions constructed without negation.

the most popular non–invasive method [24], [23] today. Non–invasive methods of data analysis aim to derive conclusions by taking into account only the given data without stringent additional model assumption. Using fewer model assumption results in more generality, wider applicability, and reduced costs and time. Non–invasive methods of data analysis use only few parameters which require only simple statistical estimation procedures. However, the generated constructs, like reducts or decision rules, should be controlled using statistical testing procedures.

Hence any non–invasive method needs to be complemented by methods for statistical significance, statistical validation, estimation of errors and model selection using only parameters supplied by the data at hand. This can be related to the ideas of statisticians looking for methods which could weaken the assumptions necessary for good estimators [39], [49].

The reader can find in literature (e.g. in [23],[24], [10], [2]), [52], [94], [95]) such methods, together with discussion of results of their applications. Among them are the following ones: data filtering based on classification error [23],[24], methods for significance testing using randomization techniques (e.g. dynamic reducts and rules [10]), model selection (e.g. by rough set entropy [23],[24]; by rough modeling allowing for generation of compact and accurate models based on, e.g., receiver operating characteristic analysis and rough modeling [2],[54] by decision rule approximation [52] - with the quality described by entropy [52], [23] and quality of classification of new objects, by reduct approximation [52] - with the quality expressed by positive region [52] and quality of classification of new objects, by boundary region thinning [52] using variable precision rough set model [137]). For more details the reader is referred to the bibliography on rough sets (see e.g. the bibliography in [94],[95]). Rough set approach backed up by non–invasive methods becomes a fully fledged instrument for data mining [24].

There are some other interesting current research topics which we can only mention. Let us observe that when defining approximation of concepts, in most cases we deal with concepts creating a partition of U^{∞}. Classifying new objects one should resolve *conflicts* between votes, coming from different approximations, for or against particular concepts. Recently a hybrid rough–neuro method has been applied for learning the strategy of conflict resolution for a given data [120]. Other current research topic deals with more complex information granules than discussed so far, defined by simple formulas, and methods for fusion of information granules to induce complex information granule approximations. The interested reader is referred to papers related to *rough mereological* approach (see e.g. [92], [97]) and papers related to spatial data mining (see e.g. [91], [113]).

In what follows we discuss in more details rough set approach vs. a general task of KDD and we refer to basic notions and examples of Chapter 1. For more information the reader is referred to [82], [94], [95] and to bibliographies in these books.

1.1 Data Representation

In the simplest case the data exploited by rough set analysis are represented in the form of attribute–value tables, see Chapter 1. Such tables are also used by Pattern Recognition, Machine Learning and KDD. These simple tables are combined when forming hierarchical or distributed data structures.

A data set can be represented by a table where each row represents, for instance, an object, a case, or an event. Every column represents an attribute, or an observation, or a property that can be measured for each object; it can also be supplied by a human expert or user. This table is called an *information system*.

The choice of attributes is subjective (they are often called *conditional attributes*) and reflects our intuition about factors that influence the classification of objects in question. The chosen attributes determine in turn primitive descriptors that provide *intensions* of primitive concepts.

In many cases the target of the classification, that is, the family of concepts to be approximated is represented by an additional attribute d called *decision*.

Information systems of this kind are called *decision systems* and they are written down as triples $\mathcal{A} = (U, A, d)$.

We try in this case to approximate concepts that are defined by the known decision. This is known in Machine Learning as *supervised learning*.

In some applications it may be necessary to work with conditional attributes which are compound in the sense that they depend on other simpler attributes which in turn depend on other attributes, etc. (e.g. this happens when the values of attributes are complex structural objects, like images or algorithms to be performed). In this case it is necessary to work with hierarchical data tables. A still more general case occurs when the data is distributed between a number of processing units (agents). We have to work then in a multi–agent environment. In such a case the process of induction of approximations to concepts is more complicated. This is due to the necessity of adding interface mechanisms that translate concepts and their similarity degrees from agent to agent. These problems are addressed by rough mereology (for references see [93]).

1.2 Indiscernibility

A decision system expresses all currently available knowledge about the objects in question. Such a table may be unnecessary large: some objects may be indiscernible or some attributes redundant.

Let $\mathcal{A} = (U, A)$ be an information system, then with any $B \subseteq A$ there is associated an equivalence relation $IND_{\mathcal{A}}(B)$:

$$IND_{\mathcal{A}}(B) = \{(x, x') \in U^2 \mid \forall a \in B \; a(x) = a(x')\}$$

$IND_{\mathcal{A}}(B)$ is called the *B-indiscernibility relation*, its classes are denoted by $[x]_B$. By X/B we denote the partition of U defined by the indiscernibility relation $IND(B)$. See Chapter 1 for more details.

1.3 Lower and upper approximation of sets, boundary regions, positive regions

We have already mentioned that vague concepts may be only approximated by crisp concepts; these approximations are recalled now.

Let $\mathcal{A} = (U, A)$ be an information system, $B \subseteq A$, and $X \subseteq U$. We can approximate X using only the information contained in B by constructing the *B-lower* and *B-upper approximations of X*, denoted $\underline{B}X$ and $\overline{B}X$ respectively, where $\underline{B}X = \{x \mid [x]_B \subseteq X\}$ and $\overline{B}X = \{x \mid [x]_B \cap X \neq \emptyset\}$.

The lower approximation induces *certain rules* while the upper approximation induces *possible rules* (i.e. rules with confidence greater than 0). The set $BN_B(X) = \overline{B}X - \underline{B}X$ is called the *B-boundary region of X* thus consisting of those objects that on the basis of the attribute set B cannot be unambiguously classified into X. The set $U - \overline{B}X$ is called the *B-outside region of X* and consists of those objects which can be with certainty classified as not belonging to X. A set is said to be *rough* (respectively, *crisp*) if the boundary region is non–empty (respectively, empty).

The following properties of approximations can easily be verified:

(1) $\underline{B}(X) \subseteq X \subseteq \overline{B}(X)$,
(2) $\underline{B}(\emptyset) = \overline{B}(\emptyset) = \emptyset, \underline{B}(U) = \overline{B}(U) = U$,
(3) $\overline{B}(X \cup Y) = \overline{B}(X) \cup \overline{B}(Y)$,
(4) $\underline{B}(X \cap Y) = \underline{B}(X) \cap \underline{B}(Y)$,
(5) $X \subseteq Y$ implies $\underline{B}(X) \subseteq \underline{B}(Y)$ and $\overline{B}(X) \subseteq \overline{B}(Y)$,
(6) $\underline{B}(X \cup Y) \supseteq \underline{B}(X) \cup \underline{B}(Y)$,
(7) $\overline{B}(X \cap Y) \subseteq \overline{B}(X) \cap \overline{B}(Y)$,
(8) $\underline{B}(-X) = -\overline{B}(X)$,
(9) $\overline{B}(-X) = -\underline{B}(X)$,
(10) $\underline{B}(\underline{B}(X)) = \overline{B}(\underline{B}(X)) = \underline{B}(X)$,
(11) $\overline{B}(\overline{B}(X)) = \underline{B}(\overline{B}(X)) = \overline{B}(X)$,

where $-X$ denotes $U - X$.

One can single out the following four basic classes of rough sets:

a) X is *roughly B-definable* iff $\underline{B}(X) \neq \emptyset$ and $\overline{B}(X) \neq U$,
b) X is *internally B-undefinable* iff $\underline{B}(X) = \emptyset$ and $\overline{B}(X) \neq U$,
c) X is *externally B-definable* iff $\underline{B}(X) \neq \emptyset$ and $\overline{B}(X) = U$,
d) X is *totally B-undefinable* iff $\underline{B}(X) = \emptyset$ and $\overline{B}(X) = U$.

These categories of vagueness have a clear intuitive meaning.

1.4 Measures of closeness of concepts

We now present some approaches to closeness measures. These are *accuracy of approximation*, *measure of positive region*, *rough membership functions* and *dependencies in a degree*. These notions are instrumental in evaluating the strength of rules and closeness of concepts as well as being applicable in determining

plausible reasoning schemes [92], [97]. Important role is also played by entropy measures (see e.g. [24] and Chapter by ŚLĘZAK).

Accuracy of approximation. A rough set X can be characterized numerically by the following coefficient

$$\alpha_B(X) = \frac{|\underline{B}(X)|}{|\overline{B}(X)|},$$

called the *accuracy of approximation*, where $|X|$ denotes the cardinality of $X \neq \emptyset$. Obviously $0 \leq \alpha_B(X) \leq 1$. If $\alpha_B(X) = 1$, X is *crisp* with respect to B (X is *exact* with respect to B), and otherwise, if $\alpha_B(X) < 1$, X is *rough* with respect to B (X is *vague* with respect to B).

Positive region and its measure. If $X_1, \ldots, X_{r(d)}$ are decision classes of $\mathcal{A}$, then the set $\underline{B}X_1 \cup \ldots \cup \underline{B}X_{r(d)}$ is called the *B–positive region of $\mathcal{A}$* and is denoted by $POS_B(d)$. The number $|POS_B(d)|/|U|$ measures closeness of the partition defined by the decision to its approximation defined by attributes from B.

Rough membership function. In classical set theory either an element belongs to a set or it does not. The corresponding membership function is the characteristic function of the set, i.e. the function takes values 1 and 0, respectively. In the case of rough sets the notion of membership is different. The *rough membership function* quantifies the degree of relative overlap between the set X and the equivalence class to which x belongs. It is defined as follows:

$$\mu_X^B(x) : U \longrightarrow [0,1] \text{ and } \mu_X^B(x) = \frac{|[x]_B \cap X|}{|[x]_B|}$$

The rough membership function can be interpreted as a frequency–based estimate of $\Pr(y \in X \mid u)$, the conditional probability that object y belongs to set X, given knowledge of the information signature $u = Inf_B(x)$ of x with respect to attributes B. The value $\mu_X^B(x)$ measures closeness of $\{y \in U : Inf_B(x) = Inf_B(y)\}$ and X.

The formulae for the lower and upper set approximations can be generalized to some arbitrary level of precision $\pi \in (\frac{1}{2}, 1]$ by means of the *variable precision rough membership* function [137] (see below).

Note that the lower and upper approximations as originally formulated are obtained as a special case with $\pi = 1.0$.

$$\underline{B}_\pi X = \{x \mid \mu_X^B(x) \geq \pi\}$$

$$\overline{B}_\pi X = \{x \mid \mu_X^B(x) > 1 - \pi\}$$

Sets of patients, events, outcomes, etc. can be approximated by variable precision rough sets with a varied precision that depends on the parameter π. Assuming data influenced by noise, one can tune the threshold π to find the "best" concept approximation. One can, e.g., start from π "close" to 1 and incrementally decrease the value of π. In each step, e.g., lower approximations of decision

classes are calculated and corresponding decision rules are induced. As lower approximations of decision classes are becoming larger when the parameter π is decreasing, induced decision rules are becoming stronger, e.g., being supported by more objects. Decrease in value of the parameter π should be stopped when the quality of new object classification by induced rules starts to decrease.

Dependencies in a degree. Another important issue in data analysis is discovering dependencies among attributes. Intuitively, a set of attributes D depends totally on a set of attributes C, denoted $C \Rightarrow D$, if all values of attributes from D are uniquely determined by values of attributes from C. In other words, D depends totally on C, if there exists a functional dependency between values of D and C. Dependency can be formally defined in the following way cf. Chapter 1.

Let D and C be subsets of A. We will say that D *depends on* C to a *degree* k $(0 \leq k \leq 1)$, denoted $C \Rightarrow_k D$, if

$$k = \gamma(C, D) = \frac{|POS_C(D)|}{|U|},$$

where $POS_C(D) = POS_C(d_D)$.
Obviously

$$\gamma(C, D) = \sum_{X \in U/D} \frac{|\underline{C}(X)|}{|U|}.$$

If $k = 1$ we say that D *depends totally* on C, and if $k < 1$, we say that D *depends partially* (to a *degree* k) on C. $\gamma(C, D)$ describes the closeness of the partition U/D and its approximation with respect to conditions in C.

The coefficient k expresses the ratio of all elements of the universe which can be properly classified to blocks of the partition U/D by employing attributes C. It will be called the *degree of the dependency.*

All the closeness measures mentioned above are constructed on the basis of the available attributes. Two important problems are the extraction of relevant parameterized closeness measures and methods of their tuning in the process of concept approximation (see section 1).

1.5 Reduct and core

In the previous section we investigated one dimension of reducing data which aimed at creating equivalence classes. The gain is apparent: only one element of the equivalence class is needed to represent the entire class. The other dimension in reduction is to store only those attributes that suffice to preserve the chosen indiscernibility relation and, consequently, the concept approximations. The remaining attributes are redundant since their removal does not worsen the classification.

Given an information system $\mathcal{A} = (U, A)$ a *reduct* is a minimal set of attributes $B \subseteq A$ such that $IND_{\mathcal{A}}(B) = IND_{\mathcal{A}}(A)$, cf. Chapter 1. In other words, a reduct is a minimal set of attributes from A that preserves the original classification defined by the set A of attributes. Finding a minimal reduct is NP-hard

[111]. One can also show that the number of reducts of an information system with m attributes can be equal to

$$\binom{m}{\lfloor m/2 \rfloor}$$

There exist fortunately good heuristics that compute sufficiently many reducts in an often acceptable time. Boolean reasoning can be successfully applied in the task of reduct finding, cf. Chapter 1. We recall this algorithmic procedure here.

For $\mathcal{A}$ with n objects, the *discernibility matrix* of $\mathcal{A}$ is a symmetric $n \times n$ matrix with entries c_{ij} as given below. Each entry consists of the set of attributes upon which objects x_i and x_j differ.

$$c_{ij} = \{a \in A \mid a(x_i) \neq a(x_j)\} \quad i, j = 1, ..., n$$

A *discernibility function* $f_{\mathcal{A}}$ for an information system $\mathcal{A}$ is a Boolean function of m Boolean variables $a_1^*, ..., a_m^*$ (corresponding to the attributes $a_1, ..., a_m$) defined below, where $c_{ij}^* = \{a^* \mid a \in c_{ij}\}$.

$$f_{\mathcal{A}}(a_1^*, ..., a_m^*) = \bigwedge \left\{ \bigvee c_{ij}^* \mid 1 \leq j \leq i \leq n, c_{ij} \neq \emptyset \right\}$$

The set of all prime implicants of $f_{\mathcal{A}}$ determines the set of all reducts of $\mathcal{A}$. In the sequel we will write a_i instead of a_i^*.

The intersection of all reducts is the so-called *core* (which may be empty).

In general, the decision is not constant on the indiscernibility classes. Let $\mathcal{A} = (U, A \cup \{d\})$ be a decision system. The *generalized decision in* $\mathcal{A}$ is the function $\partial_A : U \longrightarrow \mathcal{P}(V_d)$ defined by $\partial_A(x) = \{i \mid \exists x' \in U \; x' \; IND(A) \, x \text{ and } d(x') = i\}$. A decision system $\mathcal{A}$ is called *consistent (deterministic)*, if $|\partial_A(x)| = 1$ for any $x \in U$, otherwise $\mathcal{A}$ is *inconsistent (non-deterministic)*. Any set consisting of all objects with the same generalized decision value is called the *generalized decision class*.

It is easy to see that a decision system $\mathcal{A}$ is consistent if, and only if, $POS_A(d) = U$. Moreover, if $\partial_B = \partial_{B'}$, then $POS_B(d) = POS_{B'}(d)$ for any pair of non-empty sets $B, B' \subseteq A$. Hence the definition of a decision-relative reduct: a subset $B \subseteq A$ is a *relative reduct* if it is a minimal set such that $POS_A\{d\} = POS_B\{d\}$. Decision-relative reducts may be found from a discernibility matrix: $M^d(\mathcal{A}) = (c_{ij}^d)$ assuming $c_{ij}^d = \emptyset$ if $d(x_i) = d(x_j)$ and $c_{ij}^d = c_{ij} - \{d\}$, otherwise. Matrix $M^d(\mathcal{A})$ is called *the decision-relative discernibility matrix of* $\mathcal{A}$. Construction of *the decision-relative discernibility function* from this matrix follows the construction of the discernibility function from the discernibility matrix. It has been shown [111] that the set of *prime implicants* of $f_M^d(\mathcal{A})$ defines the set of all *decision-relative reducts* of $\mathcal{A}$.

In some applications, instead of reducts we prefer to use their approximations called α-*reducts*, where $\alpha \in [0, 1]$ is a real parameter. For a given information system $\mathcal{A} = (\mathcal{U}, \mathcal{A})$, the set of attributes $B \subset A$ is called an α-reduct in case B has a non–empty intersection with at least $\alpha \cdot 100\%$ of non–empty entries $c_{i,j}$ in the discernibility matrix of $\mathcal{A}$.

Here, it will be important to make some remarks because the most methods discussed later are based on generation of some kinds of reducts.

The discernibility matrix creates a kind of universal *board game* used to develop efficient heuristics (see e.g. [110]). High computational complexity of analyzed problems (like NP-hardness of minimal reduct computation problem [111]) is not due to their formulation using Boolean reasoning framework but it is the intrinsic property of these problems. In some sense one cannot expect that by using other formalization the computational complexity of these problems can be decreased.

One should take into account the fact that discernibility matrices are of large size for large data sets. Nevertheless, it was possible to develop efficient and high quality heuristics for quite large data sets (see e.g. papers and bibliography in [94], [95]). This was possible due to the fact that in general it is not necessary to store the whole discernibility matrix and analyze all of its entries. This follows from reasons like: (i) only some short reducts should be computed; (ii) for some kinds of reducts, like reducts relative to objects (see Section 1.6) only one column of the matrix is important; (iii) in discretization of real value attributes, some additional knowledge about the data can be used (see Section 2) in searching for relevant (for computing reducts) Boolean variables. Let us also note that our approach is strongly related to propositional reasoning [108] and further progress in propositional reasoning will bring further progress in discussed methods.

For data sets too large to be analyzed by developed heuristics, several approaches have been developed. The first one is based on decomposition of large data into regular sub-domains of size feasible for developed methods. We will shortly discuss this method later. The second one, a statistical approach, is based on different sampling strategies. Samples are analyzed using the developed strategies and stable constructs for sufficiently large number of samples are considered as relevant for the whole table. This approach has been successfully used for generating different kinds of so called *dynamic reducts* (see e.g. [10]). It yields so called *dynamic decision rules*. Experiments with different data sets have proved these methods to be promising in case of large data sets. Another interesting method (see e.g. [75]) has shown that Boolean reasoning methodology can be extended to large relational data bases. The main idea is based on observation that relevant Boolean variables for very large formula (corresponding to analyzed relational data base) can be discovered by analyzing some statistical information. This statistical information can be efficiently extracted from large data bases.

1.6 Decision rules

Reducts serve the purpose of inducing *minimal* decision rules. Any such rule contains the minimal number of descriptors in the conditional part so that their conjunction defines the largest subset of a generalized decision class (decision class, if the decision table is deterministic). Hence, information included in conditional part of any minimal rule is sufficient for prediction of the generalized decision value for all objects satisfying this part. Conditional parts of minimal

rules define neighborhoods relevant for generalized decision classes approximation. It turns out that conditional parts of minimal rules can be computed (by means of Boolean reasoning) as so called *reducts relative to objects* (see e.g. [109], [10]). Once these reducts have been computed, conditional parts of rules are easily constructed by laying the reducts over the original decision system and reading off the values. In the discussed case the generalized decision value is preserved during the reduction. One can consider stronger constraints which should be preserved. For example, in [114] the constraints are described by probability distributions corresponding to information signatures of objects cf. Chapter by ŚLĘZAK. Again, the same methodology can be used to compute *generalized reducts* corresponding to these constraints.

We recall, cf. Chapter 1, that rules are defined as follows.

Let $\mathcal{A} = (U, A, d)$ be a decision system. Atomic formulae over $B \subseteq A \cup \{d\}$ and V are expressions of the form $a = v$; they are called *descriptors* over B and V, where $a \in B$ and $v \in V_a$. The set $\mathcal{F}(B, V)$ of formulae over B and V is the least set containing all atomic formulae over B and V and closed under propositional connectives $\wedge$ (conjunction), $\vee$ (disjunction) and $\neg$ (negation).

The semantics (meaning) of the formulae is also defined recursively. For $\varphi \in \mathcal{F}(B, V)$, the meaning of φ in the decision system $\mathcal{A}$ denoted $[\varphi]_{\mathcal{A}}$ is the set of all objects in U with the property φ:

1. if φ is of the form $a = v$ then $[\varphi]_{\mathcal{A}} = \{x \in U \mid a(x) = v\}$
2. $[\varphi \wedge \varphi']_{\mathcal{A}} = [\varphi]_{\mathcal{A}} \cap [\varphi']_{\mathcal{A}}$; $[\varphi \vee \varphi']_{\mathcal{A}} = [\varphi]_{\mathcal{A}} \cup [\varphi']_{\mathcal{A}}$; $[\neg\varphi]_{\mathcal{A}} = U - [\varphi]_{\mathcal{A}}$

The set $\mathcal{F}(B, V)$ is called the set of *conditional formulae of* $\mathcal{A}$ and is denoted $\mathcal{C}(B, V)$.

A *decision rule* for $\mathcal{A}$ is any expression of the form $\varphi \Rightarrow d = v$, where $\varphi \in \mathcal{C}(B, V)$, $v \in V_d$ and $[\varphi]_{\mathcal{A}} \neq \emptyset$. Formulae φ and $d = v$ are referred to as the *predecessor* and the *successor* of the decision rule $\varphi \Rightarrow d = v$.

A decision rule $\varphi \Rightarrow d = v$ is *true* in $\mathcal{A}$ if and only if $[\varphi]_{\mathcal{A}} \subseteq [d = v]_{\mathcal{A}}$.

For a systematic overview of rule induction see the bibliography in [95].

Several numerical factors can be associated with a synthesized rule. For example, the support of a decision rule is the number of objects that match the predecessor of the rule. Various frequency-related numerical quantities may be computed from such counts.

The main challenge in inducing rules from decision systems lies in determining which attributes should be included in the conditional part of the rule. First, one can compute minimal rules. Their conditional parts describe largest object sets (definable by conjunctions of descriptors) with the same generalized decision value in a given decision system. Hence, (compare Section 1) they create largest neighborhoods still relevant for defining the decision classes (or sets of decision classes when the decision system is inconsistent). Although such minimal decision rules can be computed, this approach may result in a set of rules of not satisfactory classification quality. Such detailed rules might be over–fit and they will poorly classify new cases. Rather, shorter rules should be synthesized. Although they will not be perfect on the known cases (influenced by noise) there

is a good chance that they will be of high quality in classifying new cases. They can be constructed by computing approximations in the above mentioned sense to reducts. The quality of approximation is characterized by a degree α of approximation. This degree can be tuned to obtain relevant neighborhoods. This is again related to our general discussion in Section 1. Using reduct approximations in place of reducts, we can obtain larger neighborhoods still relevant for decision classes description in the universe U^{∞}. Approximations of reducts received by dropping some descriptors from conditional parts of minimal rules define more general neighboorhoods, not purely included in decision classes but included in them in a satisfactory degree. It means that when the received neighborhood descriptions are considered in U^{∞} they can be more relevant for decision class (concept) approximation than neighborhoods described by exact reducts, e.g. because all (or almost all) objects from the neighborhood not included in approximated decision class are those listed in U. Hence, one can expect that when by dropping a descriptor from the conditional part we receive the description of the neighborhood almost included in the approximated decision class than this descriptor is a good candidate for dropping.

For estimation of the quality of decision classes approximation global measures based on the positive region [109] or entropy [24] are used. Methods of boundary region thinning can be based e.g. on the idea of variable precision rough set model [137] (see also Section 3.1). The idea is based on an observation that neighborhoods included in decision classes in satisfactory degree can be treated as parts of the lower approximations of decision classes. Hence, lower approximations of decision classes are enlarged and decision rules generated from them are usually stronger (e.g. they are supported by more examples). The degree of inclusion is tuned experimentally to achieve e.g. high classification quality on new cases.

An other way of approaching reduct approximations is by computing reducts for random subsets of the universe of a given decision system and selecting the most stable reducts, i.e. reducts that occur in "most" subsystems. These reducts, called *dynamic reducts*, are usually inconsistent for the original table, but rules synthesized from them are more tolerant to noise and other abnormalities; rules synthesized from such reducts perform better on unseen cases since they cover most general patterns in the data (for references see the bibliography in [95] and [94]).

When a set of rules has been induced from a decision system containing a set of training examples, they can be inspected to see if they reveal any novel relationships between attributes that are worth further research. Furthermore, the rules can be applied to a set of unseen cases in order to estimate their classificatory power.

Several application schemes can be envisioned but a simple one that has proved useful in practice is the following:

1. When a *rough set classifier* (i.e. a set of decision rules together with a method for conflict resolving when they classify new cases) is confronted with a new case, then the rule set is scanned to find applicable rules, i.e. rules whose

predecessors match the case.
2. If no rule is found (i.e. no rule "fires"), the most frequent outcome in the training data is chosen.
3. If more than one rule fires, these may in turn indicate more than one possible outcome.
4. A voting process is then performed among the rules that fire in order to resolve conflicts and to rank the predicted outcomes. All votes in favor of the rule outcome are summed up and stored as its support count. Votes from all the rules are then accumulated and divided by the total number of votes in order to arrive at a numerical measure of certainty for each outcome. This measure of certainty is not really a probability but may be interpreted as an approximation to such if the model is well calibrated.

The above described strategy to resolve conflicts is the simplest one. For a systematic overview of rule application see the bibliography in [94] and [95]. Rough set methods can be used to learn from data the strategy for conflict resolving between decision rules when they are classifying new objects.

Several methods based on rough set methods have been developed to deal with large data tables, e.g. to generate strong decision rules for them. We will discuss one of these methods based on decomposition of tables by using patterns, called *templates*, see Chapter 1, describing regular sub–domains of the universe (e.g. they describe a large number of customers having a large number of common features).

Let $\mathcal{A} = (\mathcal{U}, \mathcal{A})$ be an information system. The notion of a descriptor can be generalized by using terms of the form $(a \in S)$, where $S \subseteq V_a$ is a set of values. By a *template* we mean the conjunction of descriptors, i.e. $\mathbf{T} = D_1 \wedge D_2 \wedge ... \wedge D_m$, where $D_1, ...D_m$ are either simple or generalized descriptors. We denote by $length(\mathbf{T})$ the number of descriptors in $\mathbf{T}$.

An object $u \in U$ is satisfying the template $\mathbf{T} = (a_{i_1} = v_1) \wedge ... \wedge (a_{i_m} = v_m)$ if and only if $\forall_j a_{i_j}(u) = v_j$. Hence the template $\mathbf{T}$ describes the set of objects having the common property: "*the values of attributes* $a_{j_1}, ..., a_{j_m}$ *on these objects are equal to* $v_1, ..., v_m$*, respectively*".

The *support* of $\mathbf{T}$ is defined by $support(\mathbf{T}) = |\{u \in U : u \text{ satisfies } \mathbf{T}\}|$. Long templates with a large support are preferred in many Data Mining tasks. We consider several quality functions which can be used to compare templates. The first function is defined by $quality_1(\mathbf{T}) = support(\mathbf{T}) + length(\mathbf{T})$. The second can be defined by $quality_2(\mathbf{T}) = support(\mathbf{T}) \times length(\mathbf{T})$.

Let us consider the following problems [76], see Chapter by HOA SINH NGUYEN:

1. **Optimal Template Support (OTS) Problem:**
   Instance: Information system $\mathcal{A} = (\mathcal{A}, \mathcal{U})$, and a positive integer L.
   Question: Find a template $\mathbf{T}$ with the length L and the maximal support.
2. **Optimal Template Quality (OTQ) Problem:**
   Instance: An information system $\mathcal{A} = (\mathcal{U}, \mathcal{A})$,
   Question: Find a template for $\mathcal{A}$ with optimal quality.

In [76] it has been proved that the optimal support problem (**OPT**) is NP-hard. The second problem is NP-hard with respect to $quality_1(\mathbf{T})$ and it is not known if this problem is NP-hard in case of $quality_2(\mathbf{T})$.

Large templates can be found quite efficiently by *Aprori* algorithms and its modifications (see [4, 136]). Some other methods for large template generation have been proposed (see e.g. [76]).

Templates extracted from data are used to decompose large data tables. In consequence the decision tree is built with internal nodes labeled by (extracted from data) templates, and edges outgoing from them labeled by 0 (false) and 1 (true). Any leaf is labeled by a sub–table (sub–domain) consisting of all objects from the original table matching all templates or their complements appearing on the path from the root of the tree to the leaf. The process of decomposition is continued until sub–tables attached to leaves can be efficiently analyzed by existing algorithms (e.g. decision rules for them can be generated efficiently) based on rough set methods.The reported experiments are showing that such decomposition returns many interesting regular sub–domains (patterns) of the large data table in which the decision classes (concepts) can be approximated with high quality (for references see [76], [79], [94] and [95]).

It is also possible to search for patterns that are almost included in the decision classes defining default rules [71]. For a presentation of default rules see the bibliography in [94] and [95].

2 Preprocessing

The rough set community have been committed to constructing efficient algorithms for (new) feature extraction. Rough set methods combined with Boolean reasoning [13] lead to several successful approaches to feature extraction. The most successful methods are:

- discretization techniques,
- methods of partitioning of nominal attribute value sets and
- combinations of the above methods.

Searching for new features expressed by multi-modal formulae can be mentioned here. Structural objects can be interpreted as models (so called Kripke models) of such formulas and the problem of searching for relevant features reduces to construction of multi-modal formulas expressing properties of the structural objects discerning objects or sets of objects [81].

For more details the reader is referred to the bibliography in [95].

2.1 Feature extraction: Discretization and symbolic attribute value grouping

Non-categorical attributes must be discretized in a pre–processing step. The discretization step determines how coarsely we want to view the world. Discretization, cf. Chapter 1, is a step that is not specific to the rough set approach.

A majority of rule or tree induction algorithms require it in order to perform well. The search for appropriate cutoff points can be reduced to search for prime implicants of an appropriately constructed Boolean function.

There are two reasons that we include the discussion on discretization here. First of all it is related to the general methodology of rough sets discussed at the beginning of this article. Discretization can be treated as a searching for more coarser partitions of the universe still relevant for inducing concept description of high quality. We will also show that this basic problem can be reduced to computing reducts in some appropriately defined systems. It follows that we can estimate the computational complexity of the discretization problem. Moreover, heuristics for computing reducts and prime implicants can be used here. The general heuristics can be modified to more optimal ones using a knowledge about the problem e.g. the natural order on the set of reals, etc. Discretization is only an illustrative example of many other problems with the same property.

Reported results show that discretization problems and symbolic value partition problems are of high computational complexity (i.e. NP-complete or NP-hard) which clearly justifies the importance of designing efficient heuristics. The idea of discretization is illustrated with a simple example.

Example 1. Let us consider a (consistent) decision system (see Tab. 1(a)) with two conditional attributes a and b and seven objects $u_1, ..., u_7$. Values of attributes of these objects and values of the decision d are presented in Tab. 1.

A	a	b	d
u_1	0.8	2	1
u_2	1	0.5	0
u_3	1.3	3	0
u_4	1.4	1	1
u_5	1.4	2	0
u_6	1.6	3	1
u_7	1.3	1	1

(a)

$\Longrightarrow$

$\mathbf{A^P}$	a^P	b^P	d
u_1	0	2	1
u_2	1	0	0
u_3	1	2	0
u_4	1	1	1
u_5	1	2	0
u_6	2	2	1
u_7	1	1	1

(b)

Table 1. The discretization process: (a) The original decision system $\mathcal{A}$. (b) The $\mathbf{P}$-discretization of $\mathcal{A}$, where $\mathbf{P} = \{(a, 0.9), (a, 1.5), (b, 0.75), (b, 1.5)\}$

Sets of possible values of a and b are defined by:

$$V_a = [0, 2)\,; V_b = [0, 4)\,.$$

Sets of values of a and b for objects from U are respectively given by:

$$a(U) = \{0.8, 1, 1.3, 1.4, 1.6\} \text{ and}$$
$$b(U) = \{0.5, 1, 2, 3\}$$

Discretization process produces partitions of value sets of conditional attributes into intervals in such a way that a new consistent decision system is obtained from a given consistent decision system by replacing original value of an attribute on an object in $\mathcal{A}$ with the (unique) name of the interval(s) in which this value is contained. In this way, the size of value sets of attributes may be reduced. In case a given decision system is not consistent, one can transform it into a consistent one by taking the generalized decision instead of the original decision. Discretization will then return cuts with the following property: regions bounded by them consist of objects with the same generalized decision. One can also consider *soft* (impure) cuts and induce the relevant cuts on their basis (see the bibliography in [94]).

The following intervals are obtained in our example system:

$[0.8, 1)$; $[1, 1.3)$; $[1.3, 1.4)$; $[1.4, 1.6)$ for a);

$[0.5, 1)$; $[1, 2)$; $[2, 3)$ for b).

The idea of cuts can be introduced now. Cuts are pairs (a, c) where $c \in V_a$. Our considerations are restricted to cuts defined by the middle points of the above intervals. In our example the following cuts are obtained:

$(a, 0.9)$; $(a, 1.15)$; $(a, 1.35)$; $(a, 1.5)$;

$(b, 0.75)$; $(b, 1.5)$; $(b, 2.5)$.

Any cut defines a new conditional attribute with binary values. For example, the attribute corresponding to the cut $(a, 1.2)$ is equal to 0 if $a(x) < 1.2$; otherwise it is equal to 1.

By the same token, any set P of cuts defines a new conditional attribute a_P for any a. Given a partition of the value set of a by cuts from P put the unique names for the elements of these partition.

Example 2. Let $P = \{(a, 0.9), (a, 1.5), (b, 0.75), (b, 1.5)\}$ be the set of cuts. These cuts glue together the values of a smaller then 0.9, all the values in interval $[0.9, 1.5)$ and all the values in interval $[1.5, 4)$. A similar construction can be repeated for b. The values of the new attributes a_P and b_P are shown in Tab. 1 (b).

The next natural step is to construct a set of cuts with a minimal number of elements. This may be done using Boolean reasoning.

Let $\mathcal{A} = (U, A \cup \{d\})$ be a decision system where $U = \{x_1, x_2, \ldots, x_n\}$, $A = \{a_1, \ldots, a_k\}$ and $d : U \longrightarrow \{1, \ldots, r\}$. We assume $V_a = [l_a, r_a) \subset \Re$ to be a real interval for any $a \in A$ and $\mathcal{A}$ to be a consistent decision system. Any pair (a, c) where $a \in A$ and $c \in \Re$ will be called a *cut on* V_a. Let $\mathbf{P}_a = \{[c_0^a, c_1^a), [c_1^a, c_2^a), \ldots, [c_{k_a}^a, c_{k_a+1}^a)\}$ be a partition of V_a (for $a \in A$) into subintervals for some integer k_a, where $l_a = c_0^a < c_1^a < c_2^a < \ldots < c_{k_a}^a < c_{k_a+1}^a = r_a$ and $V_a = [c_0^a, c_1^a) \cup [c_1^a, c_2^a) \cup \ldots \cup [c_{k_a}^a, c_{k_a+1}^a)$. It follows that any partition $\mathbf{P}_a$ is uniquely defined and is often identified with the set of cuts

$$\{(a, c_1^a), (a, c_2^a), \ldots, (a, c_{k_a}^a)\} \subset A \times \Re$$

.

Given $\mathcal{A} = (U, A \cup \{d\})$ any set of cuts $\mathbf{P} = \bigcup_{a \in A} \mathbf{P}_a$ defines a new decision system $\mathcal{A}^{\mathbf{P}} = (U, A^{\mathbf{P}} \cup \{d\})$ called $\mathbf{P}$-*discretization of* $\mathcal{A}$, where $A^{\mathbf{P}} = \{a^{\mathbf{P}} : a \in A\}$ and $a^{\mathbf{P}}(x) = i \Leftrightarrow a(x) \in [c_i^a, c_{i+1}^a)$ for $x \in U$ and $i \in \{0, .., k_a\}$.

Two sets of cuts $\mathbf{P}'$ and $\mathbf{P}$ are *equivalent*, written $\mathbf{P}' \equiv_{\mathbf{A}} \mathbf{P}$, iff $\mathcal{A}^{\mathbf{P}} = \mathcal{A}^{\mathbf{P}'}$. The equivalence relation $\equiv_{\mathcal{A}}$ has a finite number of equivalence classes. Equivalent families of partitions will be not discerned in the sequel.

The set of cuts $\mathbf{P}$ is called $\mathcal{A}$-*consistent* if $\partial_A = \partial_{A^{\mathbf{P}}}$, where ∂_A and $\partial_{A^{\mathbf{P}}}$ are generalized decisions of $\mathcal{A}$ and $\mathcal{A}^{\mathbf{P}}$, respectively. An $\mathcal{A}$-consistent set of cuts $\mathbf{P}^{irr}$ is $\mathcal{A}$-*irreducible* if $\mathbf{P}$ is not $\mathcal{A}$-consistent for any $\mathbf{P} \subset \mathbf{P}^{irr}$. The $\mathcal{A}$-consistent set of cuts $\mathbf{P}^{opt}$ is $\mathcal{A}$-*optimal* if $card\,(\mathbf{P}^{opt}) \leq card\,(\mathbf{P})$ for any $\mathcal{A}$-consistent set of cuts $\mathbf{P}$.

It can be shown that the decision problem of checking if for a given decision system $\mathcal{A}$ and an integer k there exists an irreducible set of cuts $\mathbf{P}$ in $\mathcal{A}$ such that $card(\mathbf{P}) < k$ ($k-$**minimal partition problem**) is NP-complete. The problem of searching for an optimal set of cuts $\mathbf{P}$ in a given decision system $\mathcal{A}$ (**optimal partition problem**) is NP-hard, see Chapter by HOA SINH NGUYEN.

Despite these complexity bounds it is possible to devise efficient heuristics that return semi–minimal sets of cuts. Heuristics based on Johnson's strategy look for a cut discerning a maximal number of object pairs and eliminate all already discerned object pairs. This procedure is repeated until all object pairs to be discerned are discerned. It is interesting to note that this can be realized by computing the minimal relative reduct of the corresponding decision system. The *"MD heuristic"* searches for a cut with a maximal number of object pairs discerned by this cut. The idea is analogous to Johnson's approximation algorithm. It may be formulated as follows:

ALGORITHM: MD-heuristics (A semi–optimal family of partitions **)**

Step 1. *Construct table* $\mathcal{A}^* = (U^*, A^* \cup \{d\})$ *from* $\mathcal{A} = (U, A \cup \{d\})$ *where* U^* *is the set of pairs* (x, y) *of objects to be discerned by* d *and* A^* *consists of attribute* c^* *for any cut* c *and* c^* *is defined by* $c^*(x, y) = 1$ *if and only if* c *discerns* x *and* y *(i.e.,* x, y *are in different half-spaces defined by* c*); set* $\mathcal{B} = \mathcal{A}^*$;

Step 2. *Choose a column from* $\mathcal{B}$ *with the maximal number of occurrences of 1's;*

Step 3. *Delete from* $\mathcal{B}$ *the column chosen in Step 2 and all rows marked with 1 in this column;*

Step 4. *If* $\mathcal{B}$ *is non-empty then go to Step 2 else Stop.*

This algorithm searches for a cut which discerns the largest number of pairs of objects (MD-heuristics). Then the cut c is moved from A^* to the resulting set of cuts $\mathbf{P}$; and all pairs of objects discerned by c are removed from U^*. The algorithm continues until U^* becomes empty.

Let n be the number of objects and let k be the number of attributes of decision system $\mathcal{A}$. The following inequalities hold: $card\,(A^*) \leq (n-1)\,k$ and $card\,(U^*) \leq \frac{n(n-1)}{2}$. It is easy to observe that for any cut $c \in A^*$ $O\left(n^2\right)$ steps are required in order to find the number of all pairs of objects discerned by c. A straightforward realization of this algorithm therefore requires $O\left(kn^2\right)$ of

memory space and $O(kn^3)$ steps in order to determine one *cut*. This approach is clearly impractical. However, it is possible to observe that in the process of searching for the set of pairs of objects discerned by currently analyzed cut from an increasing sequence of cuts one can use information about such set of pairs of objects computed for the previously considered cut. The MD-heuristic using this observation [74] determines the best cut in $O(kn)$ steps using $O(kn)$ space only. This heuristic is reported to be very efficient with respect to the time necessary for decision rules generation as well as with respect to the quality of unseen object classification.

We report some results of experiments on data sets using MD–like heuristics. We would like to comment for example on the result of classification received by an application to Shuttle data (Table 3). The result concerning classification quality is the same as the best result reported in [69] but the time is of order better than for the best result from [69]. In this table we present also the results of experiments with heuristic searching for features defined by oblique hyperplanes. This has been developed using genetic algorithm allowing to tune the position of the hyperplane to get an optimal one [74]. In this way one can implement propositional reasoning using some background knowledge about the problem.

In experiments we have chosen several data tables with real value attributes from the U.C. Irvine repository. For some tables, taking into account the small number of their objects, we have adopted the approach based on five-fold cross-validation ($CV-5$). The obtained results (Table 3) can be compared with those reported in [21, 69] (Table 2). For predicting decisions on new cases we apply only decision rules generated either by the decision tree (using hyperplanes) or by rules generated in parallel with discretization.

Names	**Nr of class.**	**Train. table**	**Test. table**	**Best results**
Australian	2	690×14	CV5	85.65%
Glass	7	214×9	CV5	69.62%
Heart	2	270×13	CV5	82.59%
Iris	3	150×4	CV5	96.00%
Vehicle	4	846×19	CV5	69.86%
Diabetes	2	768×8	CV5	76.04%
SatImage	6	4436×36	2000	90.06%
Shuttle	6	43500×7	14500	99.99%

Table 2. Data tables stored in the UC Irvine Repository

For some tables the classification quality of our algorithm is better than that of the C4.5 or Naive–Bayes induction algorithms [100] even when used with different discretization methods [21, 69, 15].

Comparing this method with the other methods reported in [69], we can conclude that our algorithms have the shortest runtime and a good overall classification quality (in many cases our results were the best in comparison to many

Data tables	Diagonal cuts		Hyperplanes	
	#cuts	quality	#cuts	quality
Australian	18	79.71%	16	82.46%
Glass	14±1	67.89%	12	70.06%
Heart	11±1	79.25%	11±1	80.37%
Iris	7±2	92.70%	6±2	96.7%
Vehicle	25	59.70%	20±2	64.42%
Diabetes	20	74.24%	19	76.08%
SatImage	47	81.73%	43	82.90%
Shuttle	15	99.99%	15	99.99%

Table 3. Results of experiments on Machine Learning data.

other methods reported in literature).

We would like to stress that inducing the minimal number of the relevant cuts is equivalent to computing the minimal reduct in a decision system constructed from the discussed above system $\mathcal{A}^*$ [74]. This in turn, as we have shown, is equivalent to the problem of computing a minimal prime implicant of Boolean function. This is only illustration of a wide class of basic problems of Machine Learning, Pattern Recognition and KDD which can be reduced to problems of relevant reduct computation.

The presented approach may be extended to the case of symbolic (nominal, qualitative) attributes as well as to the case of mixed nominal and numeric attributes.

In case of symbolic value attribute (i.e. without pre–assumed order on values of given attributes) the problem of searching for new features of the form $a \in V$ is, in a sense, from practical point of view more complicated than the for real value attributes. However, it is possible to develop efficient heuristics for this case using Boolean reasoning.

Any function $P_a : V_a \to \{1, \ldots, m_a\}$ (where $m_a \leq card(V_a)$) is called *a partition on* V_{a_i}. The *rank of* P_{a_i} is the value $rank\,(P_i) = card\,(P_{a_i}\,(V_{a_i}))$. The family of partitions $\{P_a\}_{a\in B}$ is *consistent with* B ($B-consistent$) if the condition $[(u, u') \notin IND(B/\{d\})$ implies $\exists_{a\in B}[P_a(a(u)) \neq P_a(a(u'))]]$ holds for any $(u, u') \in U$. It means that if two objects u, u' are discerned by B and d, then they must be discerned by partition attributes defined by $\{P_a\}_{a\in B}$. We consider the following optimization problem

PARTITION PROBLEM: SYMBOLIC VALUE PARTITION PROBLEM:
Given a decision table $\mathcal{A}$ and a set of attributes $B \subseteq A$, search for the minimal $B-consistent$ family of partitions (i.e. such $B-consistent$ family $\{P_a\}_{a\in B}$ that $\sum_{a\in B} rank\,(P_a)$ is minimal).

To discern between pairs of objects, we will use new binary features $a_v^{v'}$ (for $v \neq v'$) defined by $a_v^{v'}(x, y) = 1$ if $a(x) = v \neq v' = a(y)$. One can apply Johnson's heuristics for the new decision table with these attributes to search for a minimal set of new attributes that discerns all pairs of objects from different

decision classes. After extracting these sets, for each attribute a_i we construct a graph $\Gamma_{a_i} = \langle V_{a_i}, E_{a_i} \rangle$ where E_{a_i} is defined as the set of all new attributes (propositional variables) found for the attribute a_i. Any vertex coloring of Γ_{a_i} defines a partition of V_{a_i}. The colorability problem is solvable in polynomial time for $k = 2$, but remains NP-complete for all $k \geq 3$. But, similarly to discretization, one can apply some efficient heuristics searching for an optimal partition.

Let us consider an example of a decision table presented in Table 1 and (a reduced form) of its discernibility matrix (Table 1).

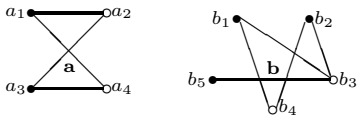
Fig. 1. The decision table and the discernibility matrix

$\mathcal{A}$	**a**	**b**	**d**
u_1	a_1	b_1	0
u_2	a_1	b_2	0
u_3	a_2	b_3	0
u_4	a_3	b_1	0
u_5	a_1	b_4	1
u_6	a_2	b_2	1
u_7	a_2	b_1	1
u_8	a_4	b_2	1
u_9	a_3	b_4	1
u_{10}	a_2	b_5	1

$\Longrightarrow$

$\mathcal{M}(\mathcal{A})$	u_1	u_2	u_3	u_4
u_5	$\mathbf{b}_{b_4}^{b_1}$	$\mathbf{b}_{b_4}^{b_2}$	$\mathbf{a}_{a_2}^{a_1}, \mathbf{b}_{b_4}^{b_3}$	$\mathbf{a}_{a_3}^{a_1}, \mathbf{b}_{b_4}^{b_1}$
u_6	$\mathbf{a}_{a_2}^{a_1}, \mathbf{b}_{b_2}^{b_1}$	$\mathbf{a}_{a_2}^{a_1}$	$\mathbf{b}_{b_3}^{b_2}$	$\mathbf{a}_{a_3}^{a_2}, \mathbf{b}_{b_2}^{b_1}$
u_7	$\mathbf{a}_{a_2}^{a_1}$	$\mathbf{a}_{a_2}^{a_1}, \mathbf{b}_{b_2}^{b_1}$	$\mathbf{b}_{b_3}^{b_1}$	$\mathbf{a}_{a_3}^{a_2}$
u_8	$\mathbf{a}_{a_4}^{a_1}, \mathbf{b}_{b_2}^{b_1}$	$\mathbf{a}_{a_4}^{a_1}$	$\mathbf{a}_{a_4}^{a_2}, \mathbf{b}_{b_3}^{b_2}$	$\mathbf{a}_{a_4}^{a_3}, \mathbf{b}_{b_2}^{b_1}$
u_9	$\mathbf{a}_{a_3}^{a_1}, \mathbf{b}_{b_4}^{b_1}$	$\mathbf{a}_{a_3}^{a_1}, \mathbf{b}_{b_4}^{b_2}$	$\mathbf{a}_{a_3}^{a_2}, \mathbf{b}_{b_4}^{b_3}$	$\mathbf{b}_{b_4}^{b_1}$
u_{10}	$\mathbf{a}_{a_2}^{a_1}, \mathbf{b}_{b_5}^{b_1}$	$\mathbf{a}_{a_2}^{a_1}, \mathbf{b}_{b_5}^{b_2}$	$\mathbf{b}_{b_5}^{b_3}$	$\mathbf{a}_{a_3}^{a_2}, \mathbf{b}_{b_5}^{b_1}$

Fig. 2. Coloring of attribute value graphs and the reduced table.



$\Longrightarrow$

$\mathbf{a}^{P_a}$	$\mathbf{b}^{P_b}$	**d**
1	**1**	0
2	**2**	0
1	**2**	1
2	**1**	1

From the Boolean function $f_{\mathcal{A}}$ with Boolean variables of the form $a_{v_1}^{v_2}$ one can find the shortest prime implicant: $\mathbf{a}_{a_2}^{a_1} \wedge \mathbf{a}_{a_3}^{a_2} \wedge \mathbf{a}_{a_4}^{a_1} \wedge \mathbf{a}_{a_4}^{a_3} \wedge \mathbf{b}_{b_4}^{b_1} \wedge \mathbf{b}_{b_4}^{b_2} \wedge \mathbf{b}_{b_3}^{b_2} \wedge \mathbf{b}_{b_3}^{b_1} \wedge \mathbf{b}_{b_5}^{b_3}$ which can be treated as graphs presented in the Figure 2.

We can color vertices of those graphs as shown in Figure 2. Colors are corresponding to partitions:

$$P_{\mathbf{a}}(a_1) = P_{\mathbf{a}}(a_3) = 1; \quad P_{\mathbf{a}}(a_2) = P_{\mathbf{a}}(a_4) = 2$$
$$P_{\mathbf{b}}(b_1) = P_{\mathbf{b}}(b_2) = P_{\mathbf{b}}(b_5) = 1; \quad P_{\mathbf{b}}(b_3) = P_{\mathbf{b}}(b_4) = 2.$$

At the same time one can construct the new decision table (Table 2).

One can extend the presented approach (see e.g. [77]) to the case when in a given decision system nominal and numeric attribute appear. The received heuristics are of very good quality.

Experiments for classification methods (see [77]) have been carried over decision systems using two techniques called *train-and-test* and *n-fold-cross-validation*. In Table 4 some results of experiments obtained by testing the proposed methods MD (using only discretization based on MD-heurisctics using the Johnson approximation strategy) and MD-G (using discretization and symbolic value grouping) for classification quality on well known data tables from the "UC Irvine repository" are shown. The results reported in [30] are summarized in columns labeled by S-ID3 and C4.5 in Table 4). It is interesting to compare those results with regard both to the classification quality. Let us note that the heuristics MD and MD-G are also very efficient with respect to the time complexity.

Names of Tables	Classification accuracies			
	S-ID3	C4.5	MD	MD-G
Australian	78.26	85.36	83.69	84.49
Breast (L)	62.07	71.00	69.95	69.95
Diabetes	66.23	70.84	71.09	76.17
Glass	62.79	65.89	66.41	69.79
Heart	77.78	77.04	77.04	81.11
Iris	96.67	94.67	95.33	96.67
Lympho	73.33	77.01	71.93	82.02
Monk-1	81.25	75.70	100	93.05
Monk-2	69.91	65.00	99.07	99.07
Monk-3	90.28	97.20	93.51	94.00
Soybean	100	95.56	100	100
TicTacToe	84.38	84.02	97.7	97.70
Average	78.58	79.94	85.48	87.00

Table 4. Quality comparison of various decision tree methods. Abbreviations: MD: MD-heuristics; MD-G: MD-heuristics with symbolic value partition

In the case of real value attributes one can search for features in the feature set that contains the characteristic functions of half–spaces determined by hyperplanes or parts of spaces defined by more complex surfaces in multi–dimensional spaces.

Genetic algorithms have been applied in searching for semi–optimal hyperplanes. The reported results are showing substantial increase in the quality of classification of unseen objects but at the cost of increased time for searching for the semi-optimal hyperplane.

2.2 Feature selection

Selection of relevant features is an important problem and has been extensively studied in Machine Learning and Pattern Recognition (see e.g. [70]). It is also a very active research area in the rough set community.

One of the first ideas [86] was to consider the *core* of the reduct set of the information system $\mathcal{A}$ as the source of relevant features. One can observe that relevant feature sets, in a sense used by the machine learning community, can be interpreted in most cases as the decision–relative reducts of decision systems obtained by adding appropriately constructed decisions to a given information system.

Another approach is related to dynamic reducts (for references see e.g. [94])cf. Chapter by BAZAN et AL. Attributes are considered relevant if they belong to dynamic reducts with a sufficiently high stability coefficient, i.e., they appear with sufficiently high frequency in random samples of a given information system. Several experiments (see [94]) show that the set of decision rules based on such attributes is much smaller than the set of all decision rules. At the same time the quality of classification of new objects increases or does not change if one only considers rules constructed over such relevant features.

Another possibility is to consider as relevant the features that come from approximate reducts of sufficiently high quality.

The idea of attribute reduction can be generalized by introducing a concept of *significance of attributes* which enables to evaluate attributes not only in the two–valued scale *dispensable – indispensable* but also in the multi–value case by assigning to an attribute a real number from the interval [0,1] that expresses the importance of an attribute in the information table.

Significance of an attribute can be evaluated by measuring the effect of removing the attribute from an information table.

Let C and D be sets of condition and decision attributes, respectively, and let $a \in C$ be a condition attribute. It was shown previously that the number $\gamma(C, D)$ expresses the degree of dependency between attributes C and D, or the accuracy of the approximation of U/D by C. It may be now checked how the coefficient $\gamma(C, D)$ changes when attribute a is removed i.e. what is the difference between $\gamma(C, D)$ and $\gamma((C - \{a\}, D)$. The difference is normalized and the significance of attribute a is defined as

$$\sigma_{(C,D)}(a) = \frac{(\gamma(C, D) - \gamma(C - \{a\}, D))}{\gamma(C, D)} = 1 - \frac{\gamma(C - \{a\}, D)}{\gamma(C, D)},$$

Coefficient $\sigma_{C,D}(a)$ can be understood as a classification error which occurs when attribute a is dropped. The significance coefficient can be extended to sets of attributes as follows:

$$\sigma_{(C,D)}(B) = \frac{(\gamma(C, D) - \gamma(C - B, D))}{\gamma(C, D)} = 1 - \frac{\gamma(C - B, D)}{\gamma(C, D)}.$$

Any subset B of C is called an *approximate reduct* of C and the number

$$\varepsilon_{(C,D)}(B) = \frac{(\gamma(C, D) - \gamma(B, D))}{\gamma(C, D)} = 1 - \frac{\gamma(B, D)}{\gamma(C, D)},$$

is called an *error of reduct approximation*. It expresses how exactly the set of attributes B approximates the set of condition attributes C with respect to determining D.

The following equations are obvious: $\varepsilon(B) = 1 - \sigma(B)$ and $\varepsilon(B) = 1 - \varepsilon(C - B)$. For any subset B of C, we have $\varepsilon(B) \leq \varepsilon(C)$. If B is a reduct of C, then $\varepsilon(B) = 0$.

The concept of an approximate reduct (with respect to the positive region) is a generalization of the concept of a reduct that was considered previously. A minimal subset B of condition attributes C, such that $\gamma(C, D) = \gamma(B, D)$, or $\varepsilon_{(C,D)}(B) = 0$ is a reduct in the previous sense. The idea of an approximate reduct can be useful in these cases where a smaller number of condition attributes is preferred over the accuracy of classification.

Several other methods of reduct approximation based on measures different from positive region have been developed. All experiments confirm the hypothesis that by tuning the level of approximation the quality of the classification of new objects may be increased in most cases. It is important to note that it is once again possible to use Boolean reasoning to compute different types of reducts and to extract from them relevant approximations.

2.3 α-reducts and association rules

In this section we discuss the relationship between association rules [4] and approximations of reducts [109], [110], [78].

Association rules can be defined in many ways (see [4]). Here, according to our notation, association rules can be defined as implications of the form $(\mathbf{P} \Rightarrow \mathbf{Q})$, where $\mathbf{P}$ and $\mathbf{Q}$ are different simple templates, i.e. formulas of the form

$$(a_{i_1} = v_{i_1}) \wedge \ldots \wedge (a_{i_k} = v_{i_k}) \Rightarrow (a_{j_1} = v_{j_1}) \wedge \ldots \wedge (a_{j_l} = v_{j_l}) \tag{1}$$

These implications can be called *generalized association rules*, because association rules were originally defined by formulas of the form $\mathbf{P} \Rightarrow \mathbf{Q}$ where $\mathbf{P}$ and $\mathbf{Q}$ were sets of items (e.g. goods or articles in stock market) e.g. $\{A, B\} \Rightarrow \{C, D, E\}$ (see [4]). One can see that this form can be obtained from 1 by replacing values of descriptors by 1 i.e.: $(A = 1) \wedge (B = 1) \Rightarrow (C = 1) \wedge (D = 1) \wedge (E = 1)$.

Usually, for a given information table $\mathcal{A}$, the quality of the association rule $\mathcal{R} = \mathbf{P} \Rightarrow \mathbf{Q}$ can be evaluated by two measures called *support* and *confidence* with respect to $\mathcal{A}$. Support of the rule $\mathcal{R}$ is defined by the number of objects from $\mathcal{A}$ satisfying the condition $(\mathbf{P} \wedge \mathbf{Q})$ i.e.

$$support(\mathcal{R}) = support(\mathbf{P} \wedge \mathbf{Q})$$

The second measure – confidence of $\mathcal{R}$ – is the ratio between the support of $(\mathbf{P} \wedge \mathbf{Q})$ and the support of $\mathbf{P}$ i.e.

$$confidence(\mathcal{R}) = \frac{support(\mathbf{P} \wedge \mathbf{Q})}{support(\mathbf{P})}$$

The following problem has been investigated by many authors (see e.g. [4, 136])

> FOR A GIVEN INFORMATION TABLE $\mathcal{A}$, AN INTEGER s, AND A REAL NUMBER $c \in [0, 1]$, FIND AS MANY AS POSSIBLE ASSOCIATION RULES $\mathcal{R} = (\mathbf{P} \Rightarrow \mathbf{Q})$ SUCH THAT $support(\mathcal{R}) \geq s$ AND $confidence(\mathcal{R}) \geq c$

All existing association rule generation methods consist of two main steps:

1. Generate as many as possible templates $\mathbf{T} = D_1 \wedge D_2 ... \wedge D_k$ such that $support(\mathbf{T}) \geq s$ and $support(\mathbf{T} \wedge D) < s$ for any descriptor D (i.e. maximal templates among those which are supported by more than s objects).
2. For any template $\mathbf{T}$, search for a partition $\mathbf{T} = \mathbf{P} \wedge \mathbf{Q}$ such that:
   (a) $support(\mathbf{P}) < \frac{support(\mathbf{T})}{c}$
   (b) $\mathbf{P}$ is the smallest template satisfying the previous condition

We show that the second step can be solved using rough set methods and Boolean reasoning approach.

Let us assume that a template $\mathbf{T} = D_1 \wedge D_2 \wedge \ldots \wedge D_m$ supported by at least s objects, has been found. For the given confidence threshold $c \in (0; 1)$, the decomposition $\mathbf{T} = \mathbf{P} \wedge \mathbf{Q}$ is called c-irreducible if $confidence(\mathbf{P} \Rightarrow \mathbf{Q}) \geq c$ and for any decomposition $\mathbf{T} = \mathbf{P}' \wedge \mathbf{Q}'$ such that $\mathbf{P}'$ is a sub–template of $\mathbf{P}$, $confidence(\mathbf{P}' \Rightarrow \mathbf{Q}') < c$.

We show that the problem of searching for c-irreducible association rules from the given template is equivalent to the problem of searching for local α-reducts (for some α) from a decision table.

Let us define a new decision table $\mathcal{A}|_{\mathbf{T}} = (U, A|_{\mathbf{T}} \cup d)$ from the original information table $\mathcal{A}$ and the template $\mathbf{T}$ by

1. $A|_{\mathbf{T}} = \{a_{D_1}, a_{D_2}, ..., a_{D_m}\}$ is a set of attributes corresponding to the descriptors of $\mathbf{T}$ such that $a_{D_i}(u) = \begin{cases} 1 \text{ if the object } u \text{ satisfies } D_i, \\ 0 \text{ otherwise.} \end{cases}$
2. the decision attribute d determines if the object satisfies template $\mathbf{T}$ i.e. $d(u) = \begin{cases} 1 \text{ if the object } u \text{ satisfies } \mathbf{T}, \\ 0 \text{ otherwise.} \end{cases}$

The following facts [110], [78] describe the relationship between association rules and approximations of reducts.

For a given information table $\mathcal{A} = (\mathcal{U}, \mathcal{A})$, a template $\mathbf{T}$, and a set of descriptors $\mathbf{P}$, an implication $\left(\bigwedge_{D_i \in \mathbf{P}} D_i \Rightarrow \bigwedge_{D_j \notin \mathbf{P}} D_j\right)$ is

1. an 100%-*irreducible association rule* from $\mathbf{T}$ if and only if $\mathbf{P}$ is a reduct in $\mathcal{A}_{\mathbf{T}}$.
2. a c-*irreducible association rule* from $\mathbf{T}$ if and only if $\mathbf{P}$ is an α-reduct in $\mathcal{A}_{\mathbf{T}}$, where $\alpha = 1 - (\frac{1}{c} - 1)/(\frac{n}{s} - 1)$, n is the total number of objects from U, and $s = support(\mathbf{T})$.

Searching for minimal α-reducts is a well known problem in rough set theory. One can show, that the problem of searching for the shortest α-reduct is NP-hard.

The following example illustrates the main idea of our method. Let us consider the following information table $\mathcal{A}$ with 18 objects and 9 attributes.

Assume that the template

$$\mathbf{T} = (a_1 = 0) \wedge (a_3 = 2) \wedge (a_4 = 1) \wedge (a_6 = 0) \wedge (a_8 = 1)$$

Table 5. The example of information table $\mathcal{A}$ and template $\mathbf{T}$ support by 10 objects and the new decision table $\mathcal{A}|_{\mathbf{T}}$ constructed from $\mathcal{A}$ and template $\mathbf{T}$

$\mathcal{A}$	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
u_1	0	1	1	1	80	2	2	2	3
u_2	0	1	2	1	81	0	aa	1	aa
u_3	0	2	2	1	82	0	aa	1	aa
u_4	0	1	2	1	80	0	aa	1	aa
u_5	1	1	2	2	81	1	aa	1	aa
u_6	0	2	1	2	81	1	aa	1	aa
u_7	1	2	1	2	83	1	aa	1	aa
u_8	0	2	2	1	81	0	aa	1	aa
u_9	0	1	2	1	82	0	aa	1	aa
u_{10}	0	3	2	1	84	0	aa	1	aa
u_{11}	0	1	3	1	80	0	aa	2	aa
u_{12}	0	2	2	2	82	0	aa	2	aa
u_{13}	0	2	2	1	81	0	aa	1	aa
u_{14}	0	3	2	2	81	2	aa	2	aa
u_{15}	0	4	2	1	82	0	aa	1	aa
u_{16}	0	3	2	1	83	0	aa	1	aa
u_{17}	0	1	2	1	84	0	aa	1	aa
u_{18}	1	2	2	1	82	0	aa	2	aa

$\mathcal{A}\|_{\mathbf{T}}$	D_1 $a_1=0$	D_2 $a_3=2$	D_3 $a_4=1$	D_4 $a_6=0$	D_5 $a_8=1$	d
u_1	1	0	1	0	0	
u_2	1	1	1	1	1	1
u_3	1	1	1	1	1	1
u_4	1	1	1	1	1	1
u_5	0	1	0	0	1	
u_6	1	0	0	0	1	
u_7	0	0	0	0	1	
u_8	1	1	1	1	1	1
u_9	1	1	1	1	1	1
u_{10}	1	1	1	1	1	1
u_{11}	1	0	1	1	0	
u_{12}	1	0	0	1	0	
u_{13}	1	1	1	1	1	1
u_{14}	1	1	0	0	0	
u_{15}	1	1	1	1	1	1
u_{16}	1	1	1	1	1	1
u_{17}	1	1	1	1	1	1
u_{18}	0	1	1	1	0	

has been extracted from the information table $\mathcal{A}$. One can see that $support(\mathbf{T}) = 10$ and $length(\mathbf{T}) = 5$. The new constructed decision table $\mathcal{A}|_{\mathbf{T}}$ is presented in Table 6. The discernibility function for $\mathcal{A}|_{\mathbf{T}}$ can be explained as follows

$$\begin{aligned} f(D_1, D_2, D_3, D_4, D_5) = &(D_2 \vee D_4 \vee D_5) \wedge (D_1 \vee D_3 \vee D_4) \wedge (D_2 \vee D_3 \vee D_4) \\ &\wedge (D_1 \vee D_2 \vee D_3 \vee D_4) \wedge (D_1 \vee D_3 \vee D_5) \\ &\wedge (D_2 \vee D_3 \vee D_5) \wedge (D_3 \vee D_4 \vee D_5) \wedge (D_1 \vee D_5) \end{aligned}$$

After simplification the condition presented in Table 6 we obtain six reducts: $f(D_1, D_2, D_3, D_4, D_5) = (D_3 \wedge D_5) \vee (D_4 \wedge D_5) \vee (D_1 \wedge D_2 \wedge D_3) \vee (D_1 \wedge D_2 \wedge D_4) \vee (D_1 \wedge D_2 \wedge D_5) \vee (D_1 \wedge D_3 \wedge D_4)$ for the decision table $\mathcal{A}|_{\mathbf{T}}$. Thus, we have found from $\mathbf{T}$ six association rules with (100%)-confidence.

For $c = 90\%$, we would like to find α-reducts for the decision table $\mathcal{A}_{\mathbf{T}}$, where $\alpha = 1 - \frac{\frac{1}{c}-1}{\frac{n}{s}-1} = 0.86$. Hence we would like to search for a set of descriptors that covers at least $\lceil (n-s)(\alpha) \rceil = \lceil 8 \cdot 0.86 \rceil = 7$ elements of discernibility matrix $\mathcal{M}(\mathcal{A}|_{\mathbf{T}})$. One can see that the following sets of descriptors: $\{D_1, D_2\}$, $\{D_1, D_3\}$, $\{D_1, D_4\}$, $\{D_1, D_5\}$, $\{D_2, D_3\}$, $\{D_2, D_5\}$, $\{D_3, D_4\}$ have nonempty intersection with exactly 7 entries of the discernibility matrix $\mathcal{M}(\mathcal{A}|_{\mathbf{T}})$. In Table 6 we present all association rules induced from those sets. Heuristics searching for α-reducts are discussed e.g. in [78].

3 Extensions of rough sets

We discuss in this section shortly two extensions of rough sets: variable precision rough set model and rough set approach based on tolerance relations but we would like to mention that many other generalizations have been investigated

Table 6. The simplified version of discernibility matrix $\mathcal{M}(\mathcal{A}|_{\mathbf{T}})$ and association rules

$\mathcal{M}(\mathcal{A}\|_{\mathbf{T}})$	u_2, u_3, u_4, u_8, u_9 $u_{10}, u_{13}, u_{15}, u_{16}, u_{17}$
u_1	$D_2 \vee D_4 \vee D_5$
u_5	$D_1 \vee D_3 \vee D_4$
u_6	$D_2 \vee D_3 \vee D_4$
u_7	$D_1 \vee D_2 \vee D_3 \vee D_4$
u_{11}	$D_1 \vee D_3 \vee D_5$
u_{12}	$D_2 \vee D_3 \vee D_5$
u_{14}	$D_3 \vee D_4 \vee D_5$
u_{18}	$D_1 \vee D_5$

$= {}_{100\%} \Longrightarrow$

$D_3 \wedge D_5 \Rightarrow D_1 \wedge D_2 \wedge D_4$
$D_4 \wedge D_5 \Rightarrow D_1 \wedge D_2 \wedge D_3$
$D_1 \wedge D_2 \wedge D_3 \Rightarrow D_4 \wedge D_5$
$D_1 \wedge D_2 \wedge D_4 \Rightarrow D_3 \wedge D_5$
$D_1 \wedge D_2 \wedge D_5 \Rightarrow D_3 \wedge D_4$
$D_1 \wedge D_3 \wedge D_4 \Rightarrow D_2 \wedge D_5$

$= {}_{90\%} \Longrightarrow$

$D_1 \wedge D_2 \Rightarrow D_3 \wedge D_4 \wedge D_5$
$D_1 \wedge D_3 \Rightarrow D_3 \wedge D_4 \wedge D_5$
$D_1 \wedge D_4 \Rightarrow D_2 \wedge D_3 \wedge D_5$
$D_1 \wedge D_5 \Rightarrow D_2 \wedge D_3 \wedge D_4$
$D_2 \wedge D_3 \Rightarrow D_1 \wedge D_4 \wedge D_5$
$D_2 \wedge D_5 \Rightarrow D_1 \wedge D_3 \wedge D_4$
$D_3 \wedge D_4 \Rightarrow D_1 \wedge D_2 \wedge D_5$

and some of them have been used for real – life data analysis. Among them are: abstract approximation spaces (see e.g. [14]); nondeterministic information systems (see e.g. [84]); extensions of rough set approach to deal with preferential ordering on attributes (criteria) in multi–criteria decision making (see e.g. [43]); extensions of rough set methods for incomplete information systems (see e.g [57] cf. Chapter by KRYSZKIEWICZ and RYBINSKI); formal languages approximations (see e.g. [83]); neighboorhood systems (see e.g. [60]); distributed systems (see e.g. [101]). For discussion of other possible extensions see [93].

3.1 The variable precision rough set model

The lower and upper approximations are just one example of possible approximations. In the terminology of Machine Learning they are approximations of subsets of objects known from the training sample. It is also desirable to approximate subsets of all objects (including also new unseen objects). The best known technique for such applications is the so–called *boundary region thinning*. It is related to the variable precision rough set approach [137]. Another technique is used in tuning of decision rules. For instance, better quality of new objects classification may be achieved by introducing some degree of inconsistency of the rules on the training objects. This technique is an analogue of the well-known techniques for decision tree pruning. These approaches can be characterized in the following way: parameterized approximations of sets are defined and better approximations of sets (or decision rules) are obtained by tuning the parameters.

3.2 Tolerance based rough set model

Tolerance relations provide an attractive and general tool for studying indiscernibility phenomena. The importance of investigations of tolerance relations had been noticed by Poincaré and Carnap.

Any tolerance relation defines a covering of the universe of objects (by neighborhoods defined by so called *tolerance classes*). Tolerance relation for objects

can be defined by similarity relation on feature (attribute) value vectors of objects cf. Chapter 1, Chapter by STEPANIUK.

A relation $\tau \subseteq X \times U$ is called a *tolerance relation* on U if (i) τ is *reflexive:* $x\tau x$ for any $x \in U$ and (ii) τ is *symmetric*: $x\tau y$ implies $y\tau x$ for any pair x, y of elements of U. The pair (U, τ) is called a *tolerance space*. It leads to a metric space with the distance function

$$d_\tau(x,y) = \min\{k : \exists_{x_0,x_1,\ldots,x_k} x_0 = x \wedge x_k = y \wedge (x_i \tau x_{i+1} \text{ for } i = 0, 1, \ldots, k-1)\}$$

Sets of the form $\tau(x) = \{y \in U : x\tau\, y\}$ are called *tolerance sets*. These sets as well as the metric above can be used to define more general approximations and their clusters. This is done by substituting tolerance classes for indiscernibility classes.

Definitions of the lower and upper approximations of sets can be easily generalized. [14] defines approximations of sets which are in some sense closer to X than the classical ones. They are defined as follows: $\tau^* X = \{x \in U : \exists y(x\tau y \& \tau(y)) \subseteq X)\}$ and $\tau_* X = \{x \in U : \forall y(x\tau y \Rightarrow \tau(y) \cap X \neq \emptyset)\}$. It is easy to check that $\underline{\tau} X \tau^* X \subseteq X \subseteq \tau_* X \subseteq \overline{\tau} X$.

It follows that in the process of learning a proper concept approximation there are more possibilities when using tolerance relations but at a greater computational cost that is due to a larger search space.

Tolerance relations can be defined for information systems or decision systems: by a *tolerance information system* we understand a triple $\mathcal{A}' = (U, A, \tau)$ where $\mathcal{A}' = (U, A)$ is an information system and τ is a tolerance relation on *information signatures* $Inf_B(x) = \{(a, a(x)) : a \in B\}$ where $x \in U$, $B \subseteq A$.

Tolerance reducts and tolerance–based decision rules can be generated by standard methods adapted to the tolerance case.

Some efficient techniques for discovery of relevant tolerances from data have been developed cf. Chapter by HOA SINH NGUYEN. For references see the bibliography in [94, 95].

3.3 Rough mereology

The approach based on inclusion in a degree has been generalized to the *rough mereological approach* (see e.g. [97], [92]) cf. Chapter 3 by POLKOWSKI and SKOWRON. The inclusion relation $x\mu_r y$ with the intended meaning *x is a part of y in a degree r* has been taken as the basic notion of the rough mereology being a generalization of the Leśniewski mereology. Rough mereology offers a methodology for synthesis and analysis of objects in distributed environment of intelligent agents, in particular, for synthesis of objects satisfying a given specification in satisfactory degree, i.e., objects sufficiently close to standard objects (prototypes) satisfying the specification. Moreover, rough mereology has been recently used for developing foundations of the *information granule calculus*, an attempt towards formalization of the Computing with Words paradigm, recently formulated by Lotfi Zadeh.

Let us also note that one of the prospects for rough mereological applications is to look for algorithmic methods of extracting logical structures from data such as, for instance, finding relational structures corresponding to relevant feature extraction, synthesizing default rules (approximate decision rules), constructing connectives for uncertainty coefficients propagation and synthesizing schemes of approximate reasoning. A progress in this direction is crucial for further development of applications, in particular, we believe it is one of the central issues for KDD [28]. Rough set approach combined with rough mereology can be treated as an inference engine for computing with words and granular computing [133], [134], [93].

4 Algebraic and Logical Aspects of Rough Sets

For any information system $\mathcal{A} = (U, A)$ one can define a family $RS(\mathcal{A})$ of *rough set representations*, i.e., pairs $(\underline{A}X, \overline{A}X)$, where $X \subseteq U$. Two questions arise immediately: (i) How to characterize the set of all rough set representations in a given information system? and (ii) What are "natural" algebraic operations on rough set representations?

A pair $(\underline{A}X, U - \overline{A}X)$ can be assigned to any rough set $(\underline{A}X, \overline{A}X)$ in $\mathcal{A}$. The following "natural" operations on those pairs of sets are: $(X_1, X_2) \wedge (Y_1, Y_2) = (X_1 \cap Y_1, X_2 \cup Y_2)$, $(X_1, X_2) \vee (Y_1, Y_2) = (X_1 \cup Y_1, X_2 \cap Y_2)$, $\sim (X_1, X_2) = (X_2, X_1)$ or $\neg(X_1, X_2) = (U - X_1, X_1)$, or $\div(X_1, X_2) = (X_2, U - X_2)$. The defined operations are not accidental: we are now very close (still the implication operation should be defined properly!) to basic varieties of abstract algebras, such as *Nelson* or *Heyting* algebras, extensively studied in connection with different logical systems. The reader can find formal analysis of the relationships of rough sets with Nelson, Heyting, Łukasiewicz, Post or double Stone algebras, in particular the representation theorems for rough sets in different classes of algebras in [94]. Let us also note that the properties of the negation operations defined above show that they correspond to the well–known negations studied in logic: strong (constructive) negation or weak (intuitionistic) negation.

Rough algebras can be derived from rough equality. Some relationships of rough algebras with many–valued logics have been shown such as, for example, soundness and completeness of Łukasiewicz's 3-valued logic with respect to rough semantics have been proven. The rough semantics defined by rough algebras is a special kind of a topological quasi–boolean algebra; relationships of rough sets with 4–valued logic have been found.

There is a number of results on logics, both propositional and predicate, that touch upon various rough set aspects. They have some new connectives (usually modal ones) reflecting different aspects of the approximations. On the semantical level they allow to express, among other possibilities, how the indiscernibility classes (or tolerance classes) are "matching" the interpretations of formulae in a given model M. For example, in the case of necessity connective the meaning $(\Box\alpha)_M$ of the formula α in the model M is the lower approximation of α_M, in case of possibility connective $(\Diamond\alpha)_M$ it is the upper approximation of α_M. Many other

connectives have been introduced and logical systems with these connectives have been characterized. For example, rough quantifiers can be defined for predicate logic. Results related to the completeness of axiomatization, decidability as well as expressibility of these logical systems are typical. The reader can find more information on rough logic in [81] and in the bibliography in [95].

It is finally worth mentioning a research direction related to the so-called rough mereological approach for an approximate synthesis of objects satisfying a given specification to a satisfactory degree. Let us note here that an interesting prospect for applied logic is to look for algorithmic methods of extracting logical structures from data. This goal is related to aims of rough mereology, several aspects in the KDD research and to the calculi on information granules and to computing with words [133] and [134]. For further references see [81], [94] and [95].

5 Applications, Case Studies, and Software Systems

There are numerous areas of successful applications of rough set software systems. Many interesting case studies are reported (for references see e.g. [94, 95], [82] and the bibliography in these books, in particular [18], [44], [54], [125], [139]).

This section lists in the alphabetical order some of the software systems for rough sets. More details can be found in [95].

- Datalogic/R, http:/ourworld.compuserve.com/homepages/reduct
- Grobian (Roughian), e-mail: I.Duentsch@ulst.ac.uk, ggediga@luce.psycho.-Uni-Osnabrueck.DE
- KDD-R: Rough Sets-Based Data Mining System, e-mail: ziarko@cs.uregina.ca
- LERS—A Knowledge Discovery System , e-mail: jerzy@eecs.ukans.edu
- PRIMEROSE, e-mail: tsumoto@computer.org
- ProbRough — A System for Probabilistic Rough Classifiers Generation, e-mail: {zpiasta,lenarcik}@sabat.tu.kielce.pl
- Rosetta Software System, http:/www.idi.ntnu.no/~aleks/rosetta/
- Rough Family - Software Implementation of the Rough Set Theory, e-mail: Roman.Slowinski@cs.put.poznan.pl, Jerzy.Stefanowski@cs.put.poznan.pl
- RSDM: Rough Sets Data Miner, e-mail: {cfbaizan, emenasalvas}@.fi.upm.es
- RoughFuzzyLab - a System for Data Mining and Rough and Fuzzy Sets Based Classification, e-mail: rswiniar@saturn.sdsu.edu
- RSL – The Rough Set Library, ftp:/ftp.ii.pw.edu.pl/pub/Rough/
- TAS: Tools for Analysis and Synthesis of Concurrent Processes using Rough Set Methods, e-mail: zsuraj@univ.rzeszow.pl
- Trance: a Tool for Rough Data Analysis, Classification, and Clustering, e-mail:wojtek@cs.vu.nl

5.1 Conclusions

Rough set theory has proved to be useful in Data Mining and Knowledge Discovery. It constitutes a sound basis for Data Mining and Knowledge Discovery

applications. The theory offers mathematical tools to discover hidden patterns in data. It identifies partial or total dependencies (i.e. cause–effect relations) in data bases, eliminates redundant data, gives approach to deal with missing data, dynamic data and others. Also methods of data mining in very large data bases using rough sets recently have been proposed and investigated.

There have been done a substantial progress in developing rough set methods for Data Mining and Knowledge Discovery (see methods and cases reported in e.g. in [16], [18], [20], [44], [51], [59], [71], [74], [82], [94], [95], [96], [139]).

New methods for extracting patterns from data (see e.g. [55], [79], [71]), [54], [90]), decomposition of decision systems (see e.g. [79]) as well as a new methodology for data mining in distributed and multi–agent systems (see e.g. [93]) have been developed.

6 Rough Logic: A Perspective on Logic in KDD

Logic understood as a study of mechanisms of inference, involving inference about knowledge from data, has evolved into many deductive schemes differing by understanding of semantics of inference $p \vdash q$. In this Chapter, basic schemes are outlined: classical calculi, many–valued logics, modal logics along with deductive mechanisms: axiomatized schemes, natural deduction, resolution, sequent calculus. Also outlined are various approaches to reasoning in inconsistent situations: belief revision, non–monotonic logics. In complex tasks of AI and KDD like pattern recognition or machine learning, inductive reasoning is more frequent in applications aimed at defining relevant concepts and dependencies among them. Examples of fuzzy logic, rough logic, mereological logic are presented.

Perception, description and analysis of real life phenomena has been a dominant feature in intellectual activity of a human being; a fortiori, this activity has been assigned to machine systems exploiting various computing paradigms. Observation of a real life phenomenon may be passive or active: by the former we mean perceiving the phenomenon and possibly rendering its impression while by the latter we mean the process in which we create tools for a quantitative description of the phenomenon in terms of measurements, recordings, expert's knowledge etc. This latter process leads to the record of the phenomenon in the form of data. Data may therefore be of many various types: numerical data, symbolic data, pattern data including time series data, etc. These types of data may be further made into more complex types e.g. arrays of numerical data or audio or video series (e.g. documentary films). The choice of a way of data collecting as well as a type of data depends on a particular problem which we are going to solve about the given phenomenon; this data elicitation process may have a great complexity and it is thoroughly studied by many authors [40]. Data elicited from a phenomenon should undergo a representation process in which they are modeled by a certain structure. This structure allows for efficient knowledge representation and reasoning about it in order to solve some queries or problems. The relationship of data and knowledge, in particular how knowledge can be

acquired from data, has attracted attention of many philosophers and logicians cf. e.g. [106].

From logical point of view, data represent a model of a phenomenon i.e. a set of entities arranged in a certain space - time structure. Clearly, a real phenomenon may have associated with it many distinct data structures.

Usually, the nature of the phenomenon suggests us certain primitive concepts i.e. sets of entities in data structure in terms of which we build more complex concepts and we carry reasoning about the phenomenon. The properties of data structures, concepts and their relationships or actions were found to be abstracted best by means of logics. A logic involves a set of formulas (well-formed expressions in a symbolic language) along with a family of relational structures (models) in which formulas are interpreted as concepts (i.e. sets of entities) of various relational complexities as well as a mechanism allowing us to reason about properties of models. The choice of the language of formulas may be critical: on one hand, this language should be expressive enough to render all essential concepts in data structures. On the other hand too expressive a language may cause too high complexity of inference process (the phenomenon of language bias in Machine Learning, Pattern Recognition, KDD [40], [68], [73], [28]). The primitive data structures constructed according to a selected way(s) of recording a phenomenon present itself as possible models for various logics. Strategies for discovery of a particular logic, relevant for problems to be solved, present a challenge for KDD.

Let us recall here a well–known logical language *DATALOG* used in relational databases (cf.[17], [126]).

Example: DATALOG

In *relational databases*, data are represented in two–dimensional tables called relations. Each row of the data table defines an instance of the relation among items occurring in this row. In DATALOG, relations are represented as *predicates*: for a relation R of arity n, we may introduce a *predicate symbol* P^n_R (often written down simply as R also) and we may define an *atom* (*an elementary formula*) $P^n_R(x_1, x_2, .., x_n)$. From atoms, more complex formulas may be constructed by means of *propositional connectives* : $\wedge$ (AND), $\vee$ (OR), $\neg$ (NOT) e.g. $P^n_R(x_1, x_2, .., x_n) \wedge P^n_S(x_1, x_2, .., x_n)$. In this way algebraic operations on relations are rendered in DATALOG in symbolic, logical form. We may have other atoms, e.g. arithmetic, like $x+y > 1$ etc. In DATALOG, we may also define *rules* as implications of the form $P^n_R(x_1, x_2, .., x_n) \longleftarrow \wedge^k_{i=1} P^{n_i}_{S_i}(x^i_{i_1}, ..., x^i_{i_{k_i}})$; rules allow for symbolical rendering of a mechanism of new relation formation.

The above aspect of DATALOG refers to its *syntax* i.e. mechanisms of formula formation. The next aspect is its *semantics* i.e. meaning, interpretation of formulas. To interpret formulas, we must *evaluate* variables $x_1, x_2, ...$; to this end, we introduce a *valuation* v as a function which assigns to each variable x_i an element $a_i = v(x_i)$ in a specified set D called the *domain of interpretation* (for instance, D may be the set of objects occurring in data tables defining relations in our relational database). We have to select *truth values*, usually as T (True),

F (False). Then we may define the meaning $[\alpha]_v$ of the atom α: $P^n_R(x_1, x_2, .., x_n)$ under the valuation v: $[\alpha]_v$=T in case $R(a_1, ..., a_n)$ holds i.e. $a_1, ..., a_n$ is a row in the data table representing R, and $[\alpha]_v$=F, otherwise. *Semantic rules* assign meanings to complex formulas e.g. $[\alpha \wedge \beta]_v$=T if and only if $[\alpha]_v$=T=$[\beta]_v$; $[\alpha \vee \beta]_v$=T if and only if $[\alpha]_v$=T or $[\beta]_v$=T; $[\neg\alpha]_v$=T if and only if $[\alpha]_v = F$.

Logical structure of DATALOG allows for procedures not allowed by the original structure of data tables by e.g. a possibility to define new concepts.

In this example the two facets of a logical system: syntax and semantics are clearly visible and the relation of the original data structure to the choice of logical language is clearly stressed.

In fitting a logic to a data structure, some important intermediate steps are to be taken:

- in the data structure certain sets of entities (concepts) and relationships among them are selected giving admissible relational structures in data;
- mechanisms of inference about properties of these admissible structures are selected (e.g. some deductive systems (see below)).

Inference mechanisms of a logic may provide us with descriptors of complex concepts hidden in data structures; for various reasons, the formulas of logic may not describe concepts in data structures exactly but approximatively only: one of reasons is that we may not know exactly the concept in question (we usually know some positive or negative examples of this concept - this is typical for Machine Learning, Data Mining and Knowledge Discovery). This makes it necessary to invoke in addition to deductive systems also inductive ones allowing us to build inference models from sets of examples. This leads to new logical systems for approximate reasoning allowing us to carry out reasoning about properties of data structures on basis of uncertain, incomplete or insufficient data [*logics for reasoning under uncertainty*]. In these logics we encounter various phenomena not experienced by classical logics like non-monotonicity, necessity of belief revision etc. (see below). When a logic is selected which approximately fits a data structure, this logic becomes an inference engine for using knowledge about a given phenomenon.

6.1 Concepts

In general, the idea of a *concept* is associated with a set of entities; given a set U of entities, we call a concept any subset of the (universe set) U. For instance, in Example on DATALOG, a concept may be any subset of the domain D of objects listed in data table. This notion of a concept is what may be called a *crisp* (*theoretical*) *concept*: the subset is understood here in the classical sense i.e. for each element of U we can decide whether it is in X or not. However, concepts in data structures are often *non-crisp* (*vague*): there are elements about which we cannot determine their membership in X with certainty. A typical example is provided by concepts expressed in natural language e.g. *high*: it may be a matter of dispute whether a man of height 175 cm is high or not. Also concepts known by examples only (observational concepts) are such. To cope with such concepts

various theories have been proposed e.g. fuzzy set theory [131], rough set theory [86], multi-valued logics [102] etc.

Concept description may be twofold: *syntactical* as well as *semantical*. In classical case, syntactical description of a concept is provided by a formula of a logic. Semantical description relative to a model (the concept meaning) is provided by the meaning of this formula i.e. a set of entities in a model which satisfy this formula. This becomes more complex in non-classical cases e.g. for fuzzy or rough concepts where models learned from training data have to be tested against new cases.

A concept may be also characterized with respect to a set of formulas: given a crisp concept X and a set F of formulas, the *intension* of X with respect to F is the subset F' of F consisting of those formulas which are satisfied by each of elements of X; assigning to a subset F' of F the family of all elements which satisfy each formula in F', we obtain a set (concept) called the *extension* of F' (in particular, we may start with X and find the extension X' of the intension F' of X; here we work in the frame of Galois connections [129]. This idea becomes more complicated in case of non-crisp concepts where a formula is satisfied by an object in a degree usually less than 1.

In many applications there is need for analyzing dynamic structures involving changes of situations (concepts) by actions; from our point of view, actions are binary relations on concepts i.e. sets of pairs (pre-condition, post-condition) [107].

Reasoning about crisp concepts may be carried out by means of *classical deductive systems*. In non-classical cases, where observational concepts are only approximated by theoretical concepts, an important additional ingredient in reasoning is provided by some measures of similarity (distance) among the concept and its approximation as well as by some mechanisms for propagation of these closeness measures or uncertainty coefficients [132]. In many applications one uses commonsense non-deductive reasoning [65] and proper knowledge representation involves ingredients from logic as well as other tools like procedural representation schemes, semantic networks or frames for representing commonsense knowledge [106].

General view on logic

In spite of many views, often contradictory, on logic and its usefulness in Artificial Intelligence, in particular in Knowledge Discovery and Data Mining, it seems that every one should agree with the statement that logic gives us a mechanism for creating aggregates (collections) of statements (regardless for now of language in which these statements are expressed) in which one statement (called the *conclusion*) is the consequence of all the remaining (called *premises*) in the sense that whenever we believe the premises we should also believe the conclusion (no matter now what we do understand by belief). The conclusion and the premises are then in the *consequence* relation. The process of passing from believed premises to the believed conclusion (*inference process*) is at heart of reasoning. Inference processes may be composed leading to chains of infer-

ences and the overall inference mechanism may be very complex. Formal logic attempts at capturing essential features and properties of inferential mechanisms applied in many various contexts. Various logics differ with respect to the language in which they construct their statements, the way in which they construct their consequence relations, and the way in which they understand the notion of belief. For instance, in classical logics (like in DATALOG, 4.1), the belief is understood as the (absolute) truth therefore consequence relations in these logics lead from true premises to the true conclusion. On the contrary, in non-classical logics, belief is understood e.g. as the probability of a statement to be true, the possibility of a statement to be true, the degree of belief in a statement to be true etc. therefore consequence relations express the degree of belief in the conclusion as a function of degrees of belief in premises.

In order to illustrate these aspects, we begin with a description of basic classical logical systems. The reader may use DATALOG as an example in the introductory part below.

Syntax, semantics

Any logic needs a language in which its statements are constructed and its inference mechanisms are represented. Hence we should define an alphabet of symbols over which well-formed expressions (formulas) of logic are to be constructed. Usually, the process of constructing formulas is of a generative character: one starts with simple (elementary, atomic) formulas and applies some generative rules for producing more complex formulas. Syntactic characteristics of a logic involves a specification of an alphabet, a class of atomic formulas as well as rules for generating formulas. This purely syntactic aspect of logic has its counterpart in the semantic aspect dealing with the meaning of formulas and the semantic aspect of consequence relations. Building semantics for a logic involves therefore a certain world (or worlds) external to the set of formulas of the logic in which we interpret formulas assigning to each of them its meaning usually being relations in this world (worlds); a good example is a relational database providing a semantic frame for logic (e.g. DATALOG). With respect to this world (worlds), we can define truth values (in general, belief degrees) of formulas. When this is done, we can study semantically acceptable consequence relations as these inference mechanisms which lead from true premises to true conclusions (respectively, from premises in which we believe in satisfactory degree to the conclusions in which we believe sufficiently) (in both cases with respect to a chosen set of worlds).

With a logic we associate therefore two basic relations: the relation of syntactic consequence denoted $\vdash$, and the relation of semantic consequence (*entailment*) denoted $\models$.

Exemplary classical logical systems (calculi)

We review now some basic logical systems. We begin with propositional logic which deals with declarative statements. In this logic we may see in the simplest form all aspects discussed above.

Propositional logic

In this logic we consider *propositions* i.e. declarative statements like *London is the capital of Great Britain* or $2 + 2 = 3$ about which we can establish with certainty whether they are true or false. The calculus of propositions is effected by means of *propositional connectives* which allow for constructions of complex propositions from simpler ones. Formally, we begin with the *alphabet* consisting of a countably many propositional symbols $p_1, p_2, ..., p_k, ...$, functor symbols $\neg$,$\Rightarrow$ and auxiliary symbols: parentheses),(,],[. An *expression* is any word over this alphabet. The set of *formulas* of propositional logic is defined as the set X such that (i) X contains all propositional symbols (ii) X with all expressions u,v contains expressions $\neg u$ and $u \Rightarrow v$ (iii) if a set Y satisfies (i), (ii) then $X \subseteq Y$. To describe X, a generative approach is also used: one sets a set $A \subset X$ of formulas called *axioms* and one specifies *derivation rules* for generating formulas from axioms. Axioms may be chosen in many distinct ways; a simple axiomatics [66] consists of the following axiom schemes (meaning that in each of these expressions we may substitute for $p, q, r, ...$ any formula and we obtain an axiom formula):

(Ax1) $(p \Rightarrow (q \Rightarrow p))$;
(Ax2) $(p \Rightarrow (q \Rightarrow r)) \Rightarrow ((p \Rightarrow q) \Rightarrow (p \Rightarrow r))$;
(Ax3) $((\neg p \Rightarrow \neg q) \Rightarrow ((\neg p \Rightarrow q) \Rightarrow r))$.

The set of derivation rules consists of a single relation on expressions called *modus ponens* MP being the set of triples of the form $(p, p \Rightarrow q, q)$ meaning that: if $p, p \Rightarrow q$ are already derived from axioms, then q is regarded also as derived. The set of *theorems* is the set of formulas which can be obtained from instances of axioms by means of applying MP a finite number of times. The basic tool in investigating syntactic properties of propositional logic is the *Herbrand deduction theorem*: *For any set of formulas* Γ *from* $\Gamma, p \vdash q$ *it follows that* $\Gamma \vdash p \Rightarrow q$.

Now, we discuss semantics of propositional logic. We evaluate formulas with respect to their truth values: *truth* (denoted by T) and *falsity* (denoted by F) assuming that functors are truth-functional i.e. they are functions on truth values and they do not depend on particular type of a formula. Under these assumptions one can characterize functors semantically by means of tables. We give the tables for $\neg, \Rightarrow$.

p	$\neg p$
0	**1**
1	**0**

Table 1. The negation functor $\neg$

p	q	$p \Rightarrow q$
0	**0**	**1**
0	**1**	**1**
1	**0**	**0**
1	**1**	**1**

Table 2. The implication functor $\Rightarrow$

Semantics of propositional logic is defined with respect to a *model* being the set of all boolean (i.e. 0, 1 - valued) functions (called valuations) on the set of propositional symbols: given a formula $\alpha(p_{i_1}, p_{i_2}, ..., p_{i_k})$ (which means that the propositional symbols $p_{i_1}, p_{i_2}, ..., p_{i_k}$ are the only variable symbols in α) and a valuation v we define the value $v(\alpha)$ with respect to α. An admissible state in the model is any valuation v such that $v(\alpha) = 1$. A formula α is true when all states are admissible i.e. $v(\alpha) = 1$ for every valuation v.

Two important properties of this deductive system are: *soundness* (meaning that every theorem is true) and *completeness* (meaning that every true formula is a theorem). It is straightforward to check the soundness of propositional logic by induction on the formula length. Less obvious is the completeness of propositional logic established first by Gödel [66]. Propositional logic is *decidable*: for each formula it is sufficient to check finitely many partial valuations restricted to the finite set of propositional symbols occurring in this formula to decide whether the formula is true. It is also *effectively axiomatizable*: for each formula one can decide in a finite number of steps whether the formula is an instance of an axiom scheme. Completeness implies *consistency*: for no formula α both α and $\neg\alpha$ can be theorems.

Let us add finally that in practical usage additional functors are introduced, familiar from DATALOG : the conjunction functor $\wedge$ defined by taking $\alpha \wedge \beta$ as a shortcut for $\neg(\alpha \Rightarrow \neg\beta)$, the disjunction functor $\vee$ defined by taking $\alpha \vee \beta$ as the shortcut for $\neg\alpha \Rightarrow \beta$ and the logical equivalence functor $\leftrightarrow$ defined by taking $\alpha \leftrightarrow \beta$ as the shortcut for $(\alpha \Rightarrow \beta) \wedge (\beta \Leftarrow \alpha)$. Truth tables for these functors follow immediately from these definitions and tables 1,2.

Propositional reasoning turned out to be very effective for solutions of many KDD problems (see Boolean reasoning in Sections 1,2) despite of the high computational complexity of the satisfiability problem of propositional calculus (it is NP-complete problem [35]).

Predicate logic

Propositional logic renders us good service by formalizing the calculus of propositions; however, in many practical situations, KDD applications including, we are concerned with properties of objects expressed as concepts i.e. sets of objects and with relations among these properties. In order to ensure the expressibility of relations e.g. inclusions (like *every ripe tomato is red*) we need to quantify statements involving object descriptors over concepts. The predicate logic is an extension of propositional logic enabling us to manipulate concept descriptors. We will write $P(x)$ to denote that the object denoted x has the property denoted P; the symbol P is a (unary) predicate symbol (cf. DATALOG). With the expression $P(x)$ we associate two expressions: $\forall x P(x)$ and $\exists x P(x)$; the symbol $\forall x$ is called the universal quantifier and $\forall x P(x)$ is read *for each object x the property P holds* and the symbol $\exists x$ is called the existential quantifier and $\exists x P(x)$ is read *there exists an object x such that P holds for x*. Predicate calculus formalizes such utterances. It will be useful to keep generality of our discussion, hence we

will give a formal analysis of deductive systems known as first order theories of which predicate calculus is a specialization. To give a formal description of first order logical calculi on lines of deductive systems, we begin with an alphabet which in general case consists of few types of symbols:

(i) *individual variables* $x_1, x_2, ..., x_k, ...$;
(ii) *individual constants* $c_1, c_2, ..., c_k, ...$;
(iii) *predicate symbols* $P^1_1, P^1_2, ..., P^k_{i_k}, ...$ where the upper index gives the arity of the predicate denoted thus;
(iv) *functional symbols* $f^1_1, f^1_2, ..., f^k_{i_k}, ...$;
(v) *symbols* $\neg$, $\Rightarrow$, $\forall x$ (where x is a variable) ,) , (.

First we define the set of *terms* by requiring it to be the set X with the properties that (i) each individual variable or constant is in X (ii) if $t_1, .., t_k$ are elements of X and $f^k_{i_k}$ is a functional symbol of arity k then $f^k_{i_k}(t_1, t_2, ..., t_k)$ is in X (iii) if Y satisfies (i), (ii) then $X \subseteq Y$.

Next, the set of *formulas* is defined as the set Z with the properties (i) for each predicate symbol $P^k_{i_k}$ and any set $\{t_1, t_2, .., t_k\}$ of terms, the expression $P^k_{i_k}(t_1, t_2, ..., t_k)$ is in Z (ii) for each pair α, β of elements of Z, the expressions $\neg\alpha, \alpha \Rightarrow \beta, \forall x\alpha$ are in Z for each variable x (iii) if Y satisfies (i), (ii) then $Z \subseteq Y$.

The existential quantifier is defined by duality clear on intuitive basis: $\exists x P(x)$ is the shortcut for $\neg\forall x\neg P(x)$. A standard distinction on occurrences of a variable x in a formula α is between free and bound occurrences which informally means that an occurrence is bound when this occurrence happens in a part of the formula (sub–formula) preceded by the quantifier sign; otherwise the occurrence is free. It is intuitively clear that a formula in which all occurrences are bound is a proposition i.e. either true or false in a given model.

To define the syntax on generative lines, one should specify the axioms of logic. Axioms of T can be divided into two groups: the first group consists of general logical axioms, the second group consists of specific axioms; when the second group is present, we speak of a *first order theory*. The axiom schemes of the first group may be chosen as follows:

- (Ax1), (Ax2), (Ax3) are axiom schemes for propositional logic.
- (Ax4) $\forall x\alpha(x) \Rightarrow \alpha(t)$ where x is a variable, t is a term and t contains no variable such that x occurs in a sub–formula quantified with respect to that variable;
- (Ax5) $\forall x(\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \forall x\beta)$ where the variable x is not free in α.

Specific axioms depend on T; for instance the theory of equivalence relation may be expressed by means of a binary predicate symbol P^2_1 and axioms
(1) $\forall x P^2_1(x, x)$;
(2) $\forall x\forall y(P^2_1(x, y) \Rightarrow P^2_1(y, x))$;
(3) $\forall x\forall y\forall z(P^2_1(x, y) \Rightarrow (P^2_1(y, z) \Rightarrow P^2_1(x, z)))$.

The set of *derivation rules* consists of two rules: modus ponens (MP) known from propositional logic and *quantification* (generalization) rule Q which is the binary relation on expressions consisting of pairs of the form $(\alpha, \forall x\alpha)$ where x is

any variable. A *theorem* of predicate logic is any formula which may be obtained from an instance of an axiom by applying a derivation rule a finite number of times. Semantics of a first order theory T is defined according to Tarski [122] as follows.

A *model* M for the theory T is a pair (D, f) where D is a set and f is an interpretation of T in D i.e. f assigns to each individual constant c an element $f(c) \in D$, to each predicate symbol P_i^k a relation $f(P_i^k)$ on D of arity k and to each functional symbol f_i^k a function $f(f_i^k)$ from D^k to D. Truth of a formula, relative to M, is defined inductively on complexity of the formula. To this end, we consider the states of the model M as sequences $\sigma = (a_i)_i$ of elements of D. Given a formula α and a state σ, we need to declare when the formula α is satisfied by σ, in symbols, $\sigma \models \alpha$. We define a map F_σ which assigns an element of D to each term of T. Individual variables are interpreted via a given state σ: $F_\sigma(x_i) = a_i$. The mapping F_σ is equal to f on individual constants. The inductive condition is as follows: if F_σ is already defined on terms $t_1, t_2, ..., t_k$ and f_i^k is a functional symbol then $F_\sigma(f_i^k(t_1, t_2, ..., t_k)) = f(f_i^k)(F_\sigma(t_1), F_\sigma(t_2), .., F_\sigma(t_k))$.

The satisfiability $\models$ is defined inductively as follows:

(i) $\sigma \models P_i^k(t_1, t_2, ..., t_k)$ if and only if $f(P_i^k)(F_\sigma(t_1), .., F_\sigma(t_k))$ holds;

(ii) $\sigma \models \neg\alpha$ if and only if it is not true that $\sigma \models \alpha$;

(iii) $\sigma \models (\alpha \Rightarrow \beta)$ if and only if $\sigma \models \alpha$ implies that $\sigma \models \beta$;

(iv) $\sigma \models \forall x \alpha(x)$ if and only if $\sigma^x \models \alpha$ for each state σ^x where (letting x to be the variable x_i) σ^x is like σ except that the i-th member of σ^x need not be equal to a_i.

These conditions allow to check for each formula whether it is *satisfied* by a given state. A formula is *true in the model* M if and only if it is satisfied by every state σ. A formula is *true* (*tautology)* if and only if it is true in every model M. Observe that a formula $\alpha(x_1, ..., x_n)$ is true if and only if its closure $\forall x_1 ... \forall x_n \alpha(x_1, ..., x_n)$ is true.

The first order theory PC without specific axioms is called the *predicate calculus*. It is obvious that properties of first order theories depend on specific axioms so we here recapitulate the facts about the predicate calculus. The soundness of predicate calculus can be easily established by structural induction: each theorem of PC is true as all instances of axiom schemes (Ax1)-(Ax3) are true and truth is preserved by derivation rules MP and Q. The important Gödel completeness theorem [41], [66] states that predicate calculus is complete: each true formula is a theorem. Decidability problems for first order theories involve questions about the formalizations of the intuitive notion of a *finite procedure* and can be best discussed in the frame of the fundamentally important first order theory of Arithmetic (cf. [66]): a predicate calculus without functional symbols and individual constants is called pure predicate calculus PP while predicate calculus with infinite sets of functional symbols and individual constants is called functional calculus PF. The classical theorem of Church [19], [66] states that both PP, PF are recursively undecidable (algorithmically unsolvable). On the other hand, many problems are recursively decidable (algorithmically solvable) however their time– or space–complexity makes them not feasible e.g. satisfiability problem for propositional calculus is NP-complete [35]. Many problems

in logic and applications in KDD are computationally hard: for those problems efficient heuristics have to be found.

6.2 Deductive systems (DS)

Here we sum up the features of deductive systems like propositional logic or predicate calculus. By a *deductive system*, we understand a tuple $(Ax, Gen, \vdash)$ where Ax is a set of *axioms* (meaning by an axiom a formula which is assumed to be well-formed and desirably true), Gen is a set of *inference* (*derivation*) *rules*, each rule R being a relation on the set of formulas and $\vdash$ is a relation on formulas such that whenever $\Gamma \vdash \alpha$ holds this means that there exists a *formal proof* of α from Γ (α is derivable from Γ) i.e. there exists a finite sequence (the formal proof) $\alpha_1, ..., \alpha_k$ such that (i) α_1 is either an axiom or an element of Γ (ii) α_k is α (iii) each α_i $(i = 2, ..., k)$ is either an axiom or is in Γ or satisfies $R(\alpha_{j_1}, ..., \alpha_{j_m}, \alpha_i)$ for some $R \in Gen$ and a subset $\{\alpha_{j_1}, ..., \alpha_{j_m}\}$ of $\{\alpha_1, ..., \alpha_{i-1}\}$. Any formula α such that $\vdash \alpha$ (meaning $\emptyset \vdash \alpha$) is said to be a *theorem of the deductive system*. From these definitions, the properties of $\vdash$ follow: (a) $\Gamma \subseteq Cn(\Gamma)$ where $Cn(\Gamma) = \{\alpha : \Gamma \vdash \alpha\}$; (b) $Cn(\Gamma) = Cn(Cn(\Gamma))$; (c) $Cn(\Gamma) \subseteq Cn(\Gamma')$ whenever $\Gamma \subseteq \Gamma'$ (the Tarski axioms for syntactic consequence [121]). Semantics of a deductive system is defined with respect to a class of specified structures called models: there exists a mechanism which for each model and each formula assigns to this formula a subset of the model domain (called the *interpretation* of the formula in the model). A formula is true with respect to a given model in case its interpretation in the model equals the model domain. A formula is *true* (is a *tautology*) in case it is true with respect to all models (in the assumed class of models).

The semantic consequence $\models$ (entailment) is defined on sets of formulas as follows: $\Gamma \models \Gamma'$ if for any model M the truth of each formula in Γ in M implies the truth of each formula from Γ' in M.

Properties of DS: soundness, consistency, completeness, decidability, expressiveness, complexity

Among properties of deductive systems there are some whose importance deserves them to be mentioned separately. The first of them is *soundness* (of axiomatization) which means that all theorems of the system are true. The dual property of *completeness* means that each true formula has a formal proof in the system. As a rule verification of soundness is straightforward while the completeness proofs are usually non-trivial. Another important property often intervening in completeness proofs is *consistency*: a set Γ of formulas is consistent if there is no formula α such that both α and its negation are derivable from Γ. Another important question is whether there exists an algorithm which for each formula can *decide* if this formula is a theorem; if yes, we say that the deductive system is decidable. In this case we may ask about the time - , space - complexity of the decidability problem. We may study complexity of other problems like *satisfiability* (whether the interpretation of the formula is non-empty). From the point of view of knowledge representation, it is important to decide what properties

can be expressed by means of formulas of the deductive system. A useful meta–rule is that the greater expressibility the greater complexity of problems about the deductive system.

Semantic tableaux, natural deduction, sequent calculus

Semantic tableaux method provides a method of determining validity in propositional or predicate calculus. A tableaux proof of $\Gamma \models \alpha$ begins with $\Gamma \cup \{\neg\alpha\}$. A tableaux proof of $\Gamma \models \alpha$ is a binary tree labeled by formulae and constructed from $\Gamma \cup \{\neg\alpha\}$ by using rules for each logical connective specifying how the tree branches. A branch of the tree closes if it contains some sentence and its negation; the tableau closes if all branches close. If the tableau closes then $\Gamma \models \alpha$ is valid. If tableau does not close and none of the rules can be applied to extend it, then $\Gamma \cup \{\neg\alpha\}$ is satisfiable and $\Gamma \models \alpha$ does not hold. For the propositional calculus, semantic tableau gives a decision procedure. In case of predicate calculus if the set $\Gamma \cup \{\neg\alpha\}$ is satisfiable i.e. $\Gamma \models \alpha$ is not true the method may never terminate (the rules for the universal quantifier can be applied repeatedly). It terminates if and only if the set $\Gamma \cup \{\neg\alpha\}$ is unsatisfiable i.e. $\Gamma \models \alpha$ holds. We say that predicate calculus is semi–decidable.

Let us mention that deduction may be formalized as so called natural deduction [38] and its form known as sequent calculus. A sequent calculus is a set of rules for transforming sequents i.e. expressions of the form $\Gamma \vdash \Delta$ where Γ, Δ are sets of formulas. Gentzen proposed [38] a set of sequent rules for classical predicate calculus.

The tableaux method can be treated as another way of writing sequent calculus derivations.

Resolution and logic programming

It is desirable from computing view point to have systems for automated deduction; the widely accepted technique for this is *resolution* due to J.A.Robinson [26]. It requires *clausal* form of formulas i.e. a conjunction of disjunctions of literals (a literal is a variable or its negation). Symbol like $\{p, q\}$ means a disjunction of literals p, q and a symbol like $\{.\}; \{..\}; ...; \{...\}$ means a conjunctions of disjunctions i.e. a clause. Resolution uses *refutational proof technique*: instead of checking validity of α it checks unsatisfiability of $\neg\alpha$; to this end $\neg\alpha$ is represented in clausal form and the resolution rule:

from clauses $a \cup p$ and $b \cup \neg p$ the clause $a \cup b$ is derived

is applied a finite number of times. Final appearance of the empty clause $\Box$ witnesses unsatisfiability of $\neg\alpha$ hence validity of α. The resolution calculus is sound and complete with respect to entailment $\models$. Resolution in predicate calculus involves unification i.e. the process of finding substitutions making two terms containing free variable identical. For extensions and refinements see [26].

Particularly important from computational point of view is *Horn clausal logic* [48] based on Horn clauses of which *Horn facts* are of the form

$\forall x_1 ... \forall x_m P^k_{i_k}(\tau_1, ..., \tau_k)$

and *Horn rules* are of the form

$\forall x_1 ... \forall x_k \alpha_1(x) \wedge .... \wedge \alpha_n(x) \Rightarrow \beta(x)$.

A set of Horn clauses is a *Horn clausal theory* T. Inferences for T are based on inference rules of the form: $\alpha_1(c) \wedge .... \wedge \alpha_n(c)/\beta(c)$ where c is a ground term i.e. term without variables corresponding to Horn rules in T. A proof $T \vdash \gamma$ is a finite sequence of inferences starting from an inference based on a fact and ending with γ. This calculus is sound and complete. Horn clausal logic can be considered as a generative device for incremental buildup of a set from Horn facts (alphabet) and Horn rules (generating rules). It has been applied in implementations of PROLOG and DATALOG in particular in logic programming [62].

The idea behind logic programming is that the logic program is a specification written as a formula in a logical language and the inference engine for the construction solution consists of a deduction system for this language. The system of deduction in logic programming is resolution. The logic programs can be of the form known as definite clause programs (a definite clause is a universally quantified disjunction of one or more literals, only one of which is negated). They are executed by adding a goal being a clause in a special form.

Semantics for logic programs can be defined by Herbrand interpretations. A Herbrand interpretation is based on the Herbrand universe i.e. the set of all ground atoms constructed from constants, function and predicate symbols in the program. The least Herbrand model of a logic program can be defined as the least fixed point of a certain function from Herbrand universe into Herbrand universe. Any predicate calculus sentence can be transformed into a set of clauses and next resolution, like in the tableau method, can be used to test, by refutation, the validity of entailment in predicate calculus.

An example of logic programming system is PROLOG (Colmerauer, 1972). More details can be found in [62]. These systems may be regarded as engines for constructing *knowledge bases* from data.

One of the main tasks of inference is to obtain a description of a target object satisfying (exactly or in satisfactory degree) a given specification (formulated in some logical language). In the *constraint programming logic* [124] the construction schemes of such objects can be extracted from logical proofs.

Theorem provers

Automated theorem proving was initiated in 1950's and by 1960 various computer programs for theorem proving were implemented (Newell, Davis and Putnam, Gilmore, Prawitz, Hao Wang) and able to prove very simple theorems. Resolution technique (J.A. Robinson, 1965) proved to be much more powerful and by now most theorem provers use resolution. In spite of progress much still remains to be done in first place in discovering proof strategies. This will need in particular to introduce some similarity measures on proofs. Moreover, KDD stimulates research towards revision of exact formal proofs by introducing instead of them schemes of approximate reasoning extracted from data [97] (see also Section 3).

Modal logic

In many applications in KDD, when our knowledge is incomplete or uncertain, e.g. in mining association rules in databases with inconsistent decision [3], we cannot have exact logical statements, but only we may express certain modalities like *property P is possible*.

Modal propositional logics deal with formalizations of phrases like *it is possible that ...*, *it is necessary that ....* These modalities are formally rendered as generalized quantifiers: $[\alpha]$ is read as *it is necessary that* α, $\langle\alpha\rangle$ is read as *it is possible that* α. These operators are related by duality: $\langle\alpha\rangle$ is the shortcut for $\neg[\neg\alpha]$. The syntax of modal calculus is defined over an alphabet much like that of propositional logic: the only addition is the introduction of modal operator symbols $[.]$ and $\langle.\rangle$. The set of formulas of modal logic is defined as the smallest set X such that (i) X contains all propositional variables (ii) with each pair α, β, X contains $\neg\alpha$, $\alpha \Rightarrow \beta$ and $[\alpha]$.

The axiomatics of modal logics depends essentially on properties of necessity which we intuitively deem as desirable; their rendering in axioms leads to various systems of modal calculi. We will briefly review the most important ones.

The simplest modal system K is obtained by adding to the axiom schemes (Ax1)-(Ax3) of propositional logic the axiom scheme (K) of the following form:

(K) $[\alpha \Rightarrow \beta] \Rightarrow ([\alpha] \Rightarrow [\beta])$.

(K) expresses our basic intuition about necessity: if both an implication and its precedent are necessarily true then the consequent should be necessarily true also. The derivation rules of modal propositional logic are: *modus ponens* MP and the *necessity rule* N which is the relation on formulas consisting of pairs of the form $(\alpha, [\alpha])$. This calculus is consistent [50]: to see it it suffices to collapse formulas of K onto formulas of propositional calculus by omitting all modal operator symbols. Then theorems of modal logic are in one–to–one correspondence with theorems of propositional calculus.

Among syntactic properties of necessity valid in K we may mention the following expressed by theorems of K: (i) $(\alpha \Rightarrow \beta) \Rightarrow ([\alpha] \Rightarrow [\beta])$; (ii) $[\alpha \vee \beta] \Rightarrow ([\alpha] \vee [\beta])$; (iii) $[\alpha \wedge \beta] \Leftrightarrow ([\alpha] \wedge [\beta])$.

Semantics of modal logic is defined as the *possible worlds* (*Kripke*) *semantics* [29]. A model for a modal logic system is a triple $M = (W, R, v)$ where W is a collection of *states* (*worlds*) and R is a binary relation on W (called *accessibility relation*); the symbol v denotes a state of the model (a valuation) i.e. the boolean function on the set of all pairs of the form (w, p) where $w \in W$ is a world and p is a propositional variable. The notion of satisfiability $M, v, w \models \alpha$ is defined by structural induction: (i) $M, v, w \models p$ if and only if $v(w, p) = 1$; (ii) $M, v, w \models \neg\alpha$ if and only if it is not true that $M, v, w \models \alpha$; (iii) $M, v, w \models \alpha \Rightarrow \beta$ if and only if either it is not true that $M, v, w \models \alpha$ or it is true that $M, v, w \models \beta$; (iv) $M, v, w \models [\alpha]$ if and only if $M, v, w_1 \models \alpha$ for each world w_1 such that $R(w, w_1)$.

A formula α is *true in the model* M if and only if $M, v, w \models \alpha$ for each world $w \in W$ and every state v; a formula is *true* if and only if it is true in each model.

It is straightforward to check that the system K is sound: all instances of axioms are true and derivation rules MP, N preserve truth of formulas. Completeness of K can be proved by e.g. the Lemmon-Scott extension of Henkin's technique of canonical models [29], [50]. Decidability of K follows from com-

pleteness and the collapse property mentioned above along with decidability of propositional calculus.

Properties of necessity axiomatized in K are by no means the only desirable; one may ask e.g. whether the property $[\alpha] \Rightarrow \alpha$ holds in K. It is easy to see that by completeness, truth of this formula requires the relation R be reflexive hence this formula is not true in general system K.

Adding to the axiom schemes of K the axiom scheme

(T) $[p] \Rightarrow p$

we obtain a new modal system T. The completeness of T can be now expressed as follows: a formula of T is a theorem of T if and only if this formula is true in all models where the accessibility relation is reflexive.

Another property of necessity is the following :

(S4) $[p] \Rightarrow [[p]]$.

Adding to axiom schemes of T the axiom scheme (S4), we obtain a new system called S4. Theorems of S4 are those formulas of K which are true in all models with the accessibility relation R reflexive and transitive.

Finally, we may consider the formula

(S5) $\langle [p] \rangle \Rightarrow [p]$.

The formula (S5) is true in all models with the accessibility relation R being an equivalence relation. As (S5) implies syntactically (S4), the system S5 obtained from T by adding the axiom scheme (S5) contains the system S4. Completeness of S5 is expressed as follows: theorems of S5 are those formulas which are true in all models with the accessibility relation R being an equivalence.

We have therefore a strictly increasing hierarchy $K, T, S4, S5$ of modal logic systems; these systems do not exhaust all possibilities [50]. Recently modal logics play an important role in many theoretical branches of Computer Science and Artificial Intelligence e.g. in formalization of reasoning by groups of intelligent agents [27]; in applicational domain, we may mention hand–written digit recognition [9] where modal formulas are used to express properties discerning between structural objects.

Temporal and dynamic logics

There are some particular contexts in which formulas of modal logic may be specialized and tailored to specific usage's. Let us mention two such cases i.e. temporal as well as dynamic logics. These logics are useful to express knowledge in a changing environment e.g. in geographic information systems (GIS') [25].

In temporal logics, the set W of possible worlds is interpreted as the set of time instants and the accessibility relation R is the precedence in time relation i.e. wRw_1 means that w precedes w_1 in time (in particular, it may happen that $w = w_1$). Clearly, R is reflexive and transitive hence this logic is of S4 type [11].

Dynamic logic is applied in the context of properties of programs [46]. In this case the set W is the set of states of an abstract computing machine. Given a program P, the modality $[.]_P$ acts on formulas describing states of the machine and its semantics is defined by the accessibility relation R_P which holds on a pair (w, w_1) if and only if an execution of P starting at w terminates at w_1; specifically, a state w satisfies the formula $[\alpha]_P$ if and only if each state w_1 such

that $R_P(w, w_1)$ satisfies α. This means that the state in which P terminates necessarily satisfies α.

Epistemic and doxastic logics

These are *logics of knowledge and belief*. Logics for reasoning about knowledge, belief, obligations, norms etc. have to deal with statements which are not merely true or false but which are known or believed etc. to be true at a moment; an additional complication is of pragmatic character: knowledge, belief etc. may be relativized to particular intelligent reasoners (agents) hence we may need also to express statements about group or common knowledge, belief etc. Modal logics have proved suitable as a general vehicle for carrying out the task of constructing such logics. These logics are useful in KDD in e.g. tasks of describing problems of distributed/many–agent nature, in building networks of reasoning agents (e.g. belief networks) etc. Epistemic logics for reasoning about knowledge [47], [130], [45] are built as modal logics with a family $K_i : i = 1, 2, ..., n$ of necessity operators, K_i interpreted as the modal operator *the agent i knows that....* Syntax of such logic is like that of modal propositional logic except for the above family K_i instead of a single [.] modal operator symbol. Formulas are defined as usual, in particular given a formula α, the expression $K_i\alpha$ is a formula, each i. We have therefore formulas like $K_iK_j\alpha$ read as *the agent i knows that the agent j knows that* α etc. Semantics is the usual Kripke semantics of possible worlds except that to accomodate all $K_i's$, a model is now a tuple $M = (W, v, R_1, ..., R_n)$ where R_i is an accessibility relation of K_i i.e. $v, w \models K_i\alpha$ if and only if $v, w' \models \alpha$ for every w' with $R_i(w, w')$. One may want to express also in this logic statements like *every agent in a group G knows that...* or *it is common knowledge among agents in G that....* This may be done by introducing additional symbols E_G, C_G for each subset $G \subseteq \{1, 2, ..., n\}$, requiring that for each formula α and each G, expressions $E_G\alpha, C_G\alpha$ be formulas and defining semantics of these formulas as follows: $v, w \models E_G\alpha$ if and only if $v, w \models K_i\alpha$ for each $i \in G$ and $v, w \models C_G\alpha$ if and only if $v, w \models K_{i_1}K_{i_2}, ..., K_{i_j}\alpha$ for each sequence $i_1i_2...i_j$ over G. In other words, the accessibility relation for E_G is $\cap\{R_i : i \in G\}$ and the accessibility relation for C_G is the transitive closure of $\{R_i : i \in G\}$. These logics are axiomatized soundly and completely exactly as modal logics of respective types; additional axiom schemes for logics endowed with operators E_G, C_G may be chosen as follows [45]: $E_Gp \leftrightarrow \wedge K_ip$; $C_Gp \leftrightarrow E_G(p \wedge C_Gp)$ along with the additional derivation rule $(p \Rightarrow E_G(\alpha \wedge \beta)) \Rightarrow (\alpha \Rightarrow C_G\beta)$. These logics are decidable (to check validity it is sufficient to examine at most 2^n worlds where n is the formula length).

Belief logics (*doxastic logics*) [36] may model belief states either as consistent sets of formulas or as sets of possible worlds. To introduce the former approach, assume that a propositional logic L is given along with its consequence relation Cn about which one usually requires *monotonicity* ($\Gamma \subseteq \Delta$ implies $Cn(\Gamma) \subseteq Cn(\Delta)$, *compactness* ($\alpha \in Cn(\Gamma)$ implies $\alpha \in Cn(\Delta)$ for a finite $\Delta \subseteq \Gamma$), *cut property* ($\alpha \in Cn(\Gamma \cup \{\beta\})$ and $\beta \in Cn(\Gamma)$ imply $\alpha \in Cn(\Gamma)$), *deduction property* ($\alpha \in Cn(\Gamma \cup \{\beta\}$ implies $\beta \Rightarrow \alpha \in Cn(\Gamma)$). Belief states are represented by sets

of formulas of L; a principal assumption may be that believing in a formula should imply believing in all consequences of this formula hence representing sets should be closed under consequence Cn. In this approach calculus of beliefs is reduced to calculus on closed sets of formulas of L. The latter approach in which belief states are represented as sets of possible worlds in which a formula is believed to be true may be shown to be equivalent to the former.

Deontic logics i.e. logics of normative concepts like obligations, prohibitions, permissions, commitments [8] are also built as modal logics. A deontic logic has an alphabet of propositional logic endowed with modal operators O, P (obligation, permission) modeled as necessity, resp. possibility. Axiomatics depends on type of modal logic one wants to obtain and it is sound and complete. These logics are decidable for reasons as above.

Para–consistent and relevant logics

Investigations on *para–consistent logics* [5], [6] have been initiated in order to challenge the logical principle saying that for any two formulas α, β it follows from $\{\alpha, \neg\alpha\}$ (syntactically or semantically) that β. Para–consistent logics are motivated by philosophical considerations as well as by computer science applications. It is quite often necessary for information systems to deal with inconsistent information because of multiple sources of information or noisy data. Another area of applications is related to *belief revision*. Mechanisms of belief revision should work on inconsistent sets of beliefs. Other applications of para–consistent logics concern theories of mathematical significance. Among systems of para–consistent logics are non–adjunctive systems (initiated by Jaśkowski's discussive or discursive logic); non-truth functional systems (initiated by da Costa); para–consistent logic generated by multi-valued logic (introduced by Asenjo).

Relevant (*relevance*) *logics* are developed as systems to avoid the so-called paradoxes of material and strict implications. In addition relevance logic is trying to avoid to infer conclusions having to do nothing with the premise. These logics were pioneered by Anderson and Belnap. Relevant logics have been used in computer science and in philosophy e.g. linear logic (discovered by Girard) is a weak relevant logic with the addition of two operators.

Conditional logics

Conditional logics [80] investigate logical properties of declarative conditional phrases of natural language : *if.... then....* The problem studied in these logics is whether *material* (*propositional*) *implication* $\Rightarrow$ represents adequately such phrases especially in case of counterfactual conditionals where the premise as well as the consequent are false. Negative answers to this question lead to various systems of conditional logic. Denoting by $\rangle$ the conditional *if... then...*, one may exploit modal logic and interpret $\alpha\rangle\beta$ as true when β is true in a world closest to world(s) in which α is true (with respect to some closeness measure) where worlds may be also constructed as closed sets of propositional logic. Accordingly, a conditional logic may be built over propositional logic by adding a new symbol $\rangle$, requiring that $\alpha\rangle\beta$ be a formula in case α, β are formulas and adding a new

derivation rule: (R_1) from $\alpha \Rightarrow \beta, \gamma\rangle\alpha$ derive $\gamma\rangle\beta$. Axiom schemes of this logic are (Ax1)-(Ax3) of propositional logic plus (Ax4) $\alpha\rangle\alpha$; (Ax5) $\alpha\rangle\beta \Rightarrow (\alpha \Rightarrow \beta)$; (Ax6) $\neg\alpha\rangle\alpha \Rightarrow (\beta\rangle\alpha)$; (Ax7) $\alpha\rangle\beta \wedge \beta\rangle\alpha \Rightarrow (\alpha\rangle\gamma \Leftrightarrow \beta\rangle\gamma)$; (Ax8) $\alpha\rangle\beta \wedge \neg(\alpha\rangle\neg\gamma) \Rightarrow (\alpha \wedge \gamma\rangle\beta)$; (Ax9) $\alpha\rangle\beta \vee \neg\alpha\rangle\beta$. This is the syntax of conditional logic C2 of Stalnaker [119]. Models of C2 are of the form of a quadruple $M = (W, R, S, E)$ where W is a set of possible worlds, R is an accessibility relation, S is a selector assigning to any formula α and any world w a world $S(\alpha, w)$ such that $S(\alpha, w) \models \alpha$ and $S(\alpha, w)$ is closest to w and E assigns to any formula its extension i.e the set of worlds in which the formula is true. There may be additional postulates on S e.g. $R(w, S(\alpha, w))$ etc. One obtains a sound and complete axiomatization of $C2$. Replacing (Ax9) by (Ax9′): $\alpha \wedge \beta \Rightarrow \alpha\rangle\beta$ one obtains conditional logic VC of Lewis [80].

Logics for dealing with inconsistent situations

When we apply logic to reason in real situations we often face the problem of dealing with inconsistencies: statements we encounter at a given moment may be inconsistent with our current knowledge. A way out of this problem has been proposed as belief revision: we have to modify our knowledge in order to remove inconsistency. A guiding principle may be the economy principle of minimal changes: when modifying our knowledge we should make changes as small as possible. Doxastic logics offer a convenient playground for treating inconsistent statements by means of logics for belief revision [22]. Main operations on belief sets (i.e. closed under Cn sets of formulas) are: -*expansion*: adding α to a set Γ results in $Cn(\Gamma \cup \{\alpha\})$; -*revision*: when an inconsistent α is added, a maximal consistent subset $\Delta \subseteq \Gamma \cup \{\alpha\}$ is selected resulting in $Cn(\Delta)$; -*contraction*: removing a set $\Delta \subseteq \Gamma$ results in a closed set $\Psi \subseteq \Gamma - \Delta$. Some postulates have been proposed [36]: denoting by $\Gamma r\alpha$ resp. $\Gamma c\alpha$ the result of revision resp. contraction of Γ relative to α, we may require that: (r1) $\Gamma r\alpha$ be closed; (r2) $\alpha \in \Gamma r\alpha$; (r3) $\Gamma r\alpha \subseteq Cn(\Gamma \cup \alpha)$; (r4) if *not* $\neg\alpha \in \Gamma$ then $Cn(\Gamma \cup \alpha) \subseteq \Gamma r\alpha$; (r5) $\Gamma r\alpha$ is the set of all formulas if and only if $\vdash \neg\alpha$; (r6) $\alpha \leftrightarrow \beta \Rightarrow \Gamma r\alpha = \Gamma r\beta$. The Levi and Harper identities: $\Gamma r_c\alpha = Cn(\Gamma c(\neg\alpha) \cup \alpha)$ resp. $\Gamma c_r\alpha = \Gamma \cap \Gamma r\alpha$ allow for defining one of these operations from the other and establish a duality: $\Gamma c_{r_c}\alpha = \Gamma c\alpha$, $\Gamma r_{c_r}\alpha = \Gamma r\alpha$.

Various heuristics have been proposed for these operations e.g. -*choice contraction function*: given Γ, α, define $Max(\Gamma, \alpha)$ as the set of maximal closed sets M such that not $\alpha \in \Gamma$ and next select a member of $Max(\Gamma, \alpha)$ as $\Gamma c\alpha$; -*meet contraction function*: take $\cup Max(\Gamma, \alpha)$ as $\Gamma c\alpha$; - *epistemic entrenchment*: assign various degrees of importance to formulas in Γ and contract starting with formulas of lowest degrees.

Belief revision logics may be perceived in a wider perspective as tools for reasoning in situations when the standard consequence fails: in presence of inconsistencies non-monotonicity may arise i.e. a greater set of premises may lead to a smaller set of consequents because of need for revision of our knowledge. Attempts at formal rendering of this phenomenon has led to non-monotonic logics [65]. Non-monotonic reasoning is central for intelligent systems dealing with

commonsense reasoning being non-monotonic.

The non-monotonic logics deal with non-monotonic consequence $\vdash_{nm}$; a general idea for rendering $\vdash_{nm}$ may be as follows: try to define $\alpha \vdash_{nm} \beta$ as holding when there exists a belief set Γ of formulas, $\alpha \vdash \beta$ and $\alpha \vdash \gamma$ for sufficiently many $\gamma \in \Gamma$.

This idea is realized in various ways: in *default logic* [98], *probabilistic logic* [1], *circumscription* [53], *auto–epistemic logic* [58]. Reiter's default logic is built over predicate calculus L by enriching it with inference rules (called *defaults*) of the form $\alpha(x); \beta(x)/\gamma(x)$ where $\alpha(x), \beta(x), \gamma(x)$ are formulas called resp. the precondition, the test condition and the consequent of the default. For a constant a, the default permits to derive $\gamma(a)$ from $\alpha(a)$ under the condition that *not* $\vdash \neg\beta(a)$; we denote this consequence by $\vdash_d$. Formally, a default theory T may be represented as a pair (K, E) where K, a background context, contains rules and E, the evidence set, contains facts. Rules in K are of two kinds: rules of L and defaults D.

Let us consider one example of characterization of non-monotonic inference $\vdash_{K,E}$ from given (K, E). It may be done by a set of postulates of which we mention: (*Defaults*) $p \vdash_d q \in D$ implies $p \vdash_{K,E} q$; (*Deduction*) $\vdash p$ implies $\vdash_{K,E} p$; (*Augmentation*) $\vdash_{K,E} p$ and $\vdash_{K,E} q$ imply $\vdash_{K,E\cup\{p\}} q$; (*Reduction*) $\vdash_{K,E} p$ and $\vdash_{K,E\cup\{p\}} q$ imply $\vdash_{K,E} q$; (*Disjunction*) $\vdash_{K,E\cup\{p\}} r$ and $\vdash_{K,E\cup\{q\}} r$ imply $\vdash_{K,E\cup\{p\vee q\}} r$. As shown in [37], these rules are sound and complete under a probabilistic interpretation of ϵ – entailment [1]. In this interpretation, probability distributions P_K ϵ-consistent with K in the sense of ϵ-entailment i.e. such that $P_K(\alpha) = 1$ for each $\alpha \in L$, $P_K(\beta|\alpha) \geq 1 - \epsilon$ and $P_K(\alpha) \geq 0$ for each rule $\alpha \vdash_d \beta$ in D (where ϵ is a fixed parameter) are considered. A proposition p is ϵ-entailed by T when for each ϵ there exists a δ such that $P_K(p|E) \geq 1 - \epsilon$ for each δ-consistent probability distribution P_K.

One of the main problems for non-monotonic logics is to define for a given set A of sentences a family $E(A)$ of all its *extensions* i.e. sets of sentences acceptable by an intelligent agent as a description of the world. Different formal attempts to solve this problem are known. Some of them are trying to implement the principle called *Closed World Assumption* (facts which are not known are false). Having the set of extensions $E(A)$ one can define the skeptical consequence relation by taking the intersection of all possible extensions of A.

Let us finally observe that e.g. classification problems can be treated as problems of constraints satisfaction with constraints specified by discernibility conditions and some optimality measures (see also Section B6). Two consistent sets describing situations (objects) are satisfying the discernibility relation if their union creates an inconsistent (or inconsistent in some degree) set.

Many valued logics

Yet another treatment of inference was proposed by Jan Łukasiewicz [64] by assigning to propositions other - than *truth* and *falsity* – logical values. In the first 3–valued logic Ł$_3$ propositions were assigned additionally the value 1/2 (*possible*); the meaning of implication $p \Rightarrow q$ was determined as $min(1, 1-v(p)+v(q))$ where

$v(p)$ is the logical value of the proposition p; similarly negation was determined by the condition $v(\neg p) = 1 - v(p)$.

The same formulas were used to define semantics of $n - valued$ Łukasiewicz logic Ł$_n$ and infinite-valued logic Ł$_\omega$ (where logical values are rational numbers from the interval [0,1]). A complete axiomatization for these logics was proposed by Wajsberg [105]. Other systems for many-valued logic were proposed by Post, Kleene and others [105], [127].

In general, one may consider e.g. Łukasiewicz implication, negation etc. as interpreted as functions of suitable arity on the interval [0,1] a fortiori truth values may be regarded as real numbers from the interval [0,1]. In real-valued logics, truth-functional definitions of propositional functors rely on real functions on [0,1] in particular on so-called $t - norms$ and $t - co - norms$ which resp. define semantics of conjunction and disjunction. Implications are usually defined by Łukasiewicz or Kleene implications or so-called *residuated implications* and *negations* are interpreted as decreasing idempotent functions on [0,1] [56]. The interest in many-valued logics grew rapidly after introduction of fuzzy logic by Lotfi A. Zadeh (see below).

Constructive approach, Intuitionism

The above logics were developed on classical principles in which one admits non-constructive proofs. This point of view has been contested by many in first place by L.E.J. Brouwer who put forth an idea of intuitionistic logic; in this logic, proof is understood as a effective construction and the inference $\alpha \Rightarrow \beta$ is true when we have a construction which transform any proof of α into a proof of β. The law of excluded middle (that any sentence is either true or false) does not hold as we may have neither any constructive proof of α nor any constructive proof of $\neg\alpha$. Accepting this point of view leads to parallel intuitionistic variants of above logics. However, the notion of *constructive* is vague; it seems that intuitionistic calculi are forms of algorithmic systems [123]. Let us note that topological semantics for intuitionistic logic has been first proposed by Tarski and Stone and another approach has been proposed by Kripke.

6.3 Inductive reasoning

Inductive reasoning (inference) can be described as an art of hypothesizing a set of premises P for a given set C of consequences with respect to a given background knowledge BK i.e. $P \cup BK \models C$ [67].

In the above scheme, the unknown element is a set P of premises but also the semantic inference $\models$ has to be specified. Contrary to logical deductive semantic consequence, the inductive inference $\models$ is not concerned with absolute truth - preserving but deals with approximations of concepts and along with mechanisms for concept approximations it should also possess mechanisms for generating degrees of closeness between any concept approximated and its approximation. These degrees may be expressed as numerical values or logical expressions. It is hardly expected that the inductive inference $\models$ may be defined abstracting from the specific background knowledge BK i.e. from the applicational context;

one should rather expect a variety of inference mechanisms dependent on the context and extracted from data by using appropriate algorithmic tools. This seems to be a challenge for further development of logic.

A general scheme for inductive reasoning may thus consist of a mechanism for primitive concept formation and a mechanism for construction of complex concepts from primitives. All concepts are of approximative character and mechanisms for their construction must rely on some measures of closeness among concepts. It is important to realize that concepts may be defined in various languages and their comparison is effected by imposing measures of closeness on their extensions.

Particular areas of importance for inductive reasoning are Machine Learning, Pattern Recognition and Knowledge Discovery in Data. Various specific approaches have been proposed for inductive (approximative) inference in these areas. In addition some universal paradigms for inductive reasoning like inductive logic programming [72] have been developed e.g. fuzzy inference, rough inference, probabilistic and statistical reasoning. In what follows we will outline the basic ideas of these approaches.

General view: experimental concepts and approximate definability

Background knowledge is often expressed by means of data tables (e.g. training and test examples in Machine Learning, Pattern Recognition, Inductive Logic Programming). These data tables contain positive as well as negative examples for concepts to be learned or recognized; most often, the given examples form a relatively small part of the concept extension so we may not learn these concepts exactly but approximatively only. In constructing approximations to concepts, the choice of a language for concept description (e.g. a language of logical (boolean) formulas) involving the choice of primitive formulas as well as inference mechanisms is very important. Finding a suitable language is itself a challenging problem in scientific discovery. Approximate definability is effected by means of some measure μ of closeness (similarity) on concept extensions; one of oftener applied measures is the Łukasiewicz measure μ_L [63] based on frequency count, $\mu_L(A, B) = card(A \cap B)/card(A)$ where A, B are concept extensions, rediscovered by Machine Learning and KDD communities recently.

Some natural constraints can be put on concept approximations (i) the extension of concept approximation should be consistent with the concept extension or, at least almost consistent i.e. consistent on training examples and having *small* error on test examples; (ii) some minimality (economy) conditions e.g. minimal length of concept description, universality of description or best adaptability. These conditions are applied in Machine Learning, Pattern Recognition and KDD. Satisfying (i) as well as (ii) may be computationally hard (finding a minimal consistent description is NP-hard for propositional logic [7], [70]) hence there are strategies for suboptimal approximations. Hence as a solution to concept approximation problem we obtain as a rule a family of concept descriptions parameterized by languages chosen or strategies applied for particular solutions. Choice of a particular solution may be motivated by ease of adaptivity, compu-

tational complexity of procedures of parameter tuning etc. Let us observe that the choice of primitive concepts is actually a choice of an initial model (relational structure) which itself is to be discovered; an essential criterium is its expresiveness for concept approximations (see also Section 1).

Relationships with Machine Learning, Pattern Recognition, Inductive Logic Programming

To illustrate the general scheme, we borrow an example from KDD. A *decision rule* may be expressed in the form: $a_1 = v_1 \wedge a_2 = v_2 \wedge \ \wedge a_k = v_k \Rightarrow d = v$ where $a_1, a_2, ..., a_k$ are features (predicates) used to build formulas expressing approximating concepts and d is a (vector of) feature(s) used in formulas discerning concepts (decisions) approximated. An *association rule* [3], [28] is a decision rule in which the concept defined by the premise of the rule approximates the concept defined by the consequent of the rule in high degree (high confidence) i.e. sufficient fraction of examples satisfying the premise satisfy the consequent as well and there is a sufficient (defined by a set threshold) number of examples supporting both the premise and the consequent. Similar ideas are exploited in Machine Learning and Pattern Recognition for estimating a strength of a rule.

The problem of selecting relevant features [73], [70] involves some searching procedures like discretization, grouping of symbolic values, clusterization, morphological filtering. These preliminary procedures define primitive concepts (features) for a given problem of concept approximation. This process leads from primitive features (variables) (e.g. real-valued features, pixel-valued features to new intrinsic features (variables) at the gain being a more compact and a more general description of concepts. Another alternative in search for features is to search for hidden features (variables) - possibly better suited for concept approximation - in terms of which one may define (possibly near-to-functionally) the existing features (variables).

6.4 Reasoning about Knowledge

Approximate reasoning about knowledge

Logics for approximate reasoning about knowledge attempt using symbolic calculi to express concepts by their approximations being extensions of formulas of these logics.

Fuzzy logic

Fuzzy logic [132] is a logic of vague concepts whose descriptors (very often in natural language) are familiar to everyone but whose extensions are subjective due to distinct understanding; Lotfi Zadeh proposed to interpret such concepts as fuzzy sets. A subset $X \subseteq U$ is a *fuzzy set* in U when the membership in X is not crisp (binary) but it is subject to gradation; formally, this is expressed by requiring the characteristic function μ_X of X to be a function from U into

the interval [0,1] (not into $\{0,1\}$ as in classical set theory); usually, a fuzzy set X is identified with its fuzzy membership function μ_X. A model for a concept X is a domain D_X of this concept along with a finite set $A_1, A_2, ..., A_k$ of its *values* (*features*) interpreted as fuzzy subsets μ_{A_i} of D_X. Fuzzy logic is built over an alphabet consisting of propositional logic symbols along with symbols for concepts and their features and a symbol ι. Elementary formulas are of the form $X\iota A$ (read X *is* A) where X is a concept symbol and A is a feature symbol of that concept. Formulas are formed from elementary formulas by means of propositional functors. Models for fuzzy logics consist of a family of domains for concepts, fuzzy membership functions of concept features and certain operations for building complex domains from the simple ones. To illustrate the workings of this mechanism, consider e.g. a formula $\alpha : X\iota A \vee Y\iota B$. Interpretations of elementary formulas $X\iota A$, $Y\iota B$ are resp. fuzzy membership functions μ_A on D_X and μ_B on D_Y. These functions have first to be lifted to the cartesian product D_X x D_Y by cylindrical extensions. The resulting functions μ'_A, μ'_B may be compared and for a chosen t -co-norm T, the meaning of α is $T(\mu'_A(x,y), \mu'_B(x,y))$. Inferences in fuzzy logic are of the form of implications: **if** $X\iota A$ **then** $Y\iota B$ interpreted as functions of the form $I((\mu'_A(x,y), \mu'_B(x,y))$ where I is a many-valued logic implication. Fuzzy logic is the underlying logic of input-output signals in fuzzy controllers [56].

Rough logic

Logic of ambiguous concepts whose extensions are defined only approximatively due to the incompleteness of knowledge was proposed by Zdzisław Pawlak [86] as *rough logic*. This logic is built over data structures known as information systems or data tables formalized as pairs (U, A) where U is a universe of objects and A is a finite set of attributes (features) where any attribute a is modelled as a mapping on U with values in a value set V_a. Rough logic is built from elementary formulas (descriptors) of the form (a, v) where $v \in V_a$ and propositional connectives. The model for this logic is the universe U and the meaning of an elementary formula (a, v) is $[(a, v)] = \{x \in U : a(x) = v\}$. Letting $[\alpha \vee \beta] = [\alpha] \cup [\beta]$, $[\alpha \wedge \beta] = [\alpha] \cap [\beta]$ and $[\neg\alpha] = U - [\alpha]$ extends the meaning to all formulas. Inference rules in rough logic are of the form : **if** α **then** β (decision rules, dependencies). True inference rules are those whose meaning is the universe U; in general, one may assign to an inference rule its truth degree defined e.g. as the Łukasiewicz measure of closeness of the extension of the premise and the extension of the consequent. *Definable* (*exact, crisp*) concepts are extensions of formulas of rough logic; other concepts are *inexact* (*rough*). Their description is approximative: for each such concept X there exist two formulas: α_{-X}, α_{+X} such that $[\alpha_{-X}] \subseteq X \subseteq [\alpha_{+X}]$ and $[\alpha_{-X}]$, $[\alpha_{+X}]$ are resp. the maximal and the minimal exact concepts with this property; these approximating concepts are called resp. the lower and the upper approximation of X. Reasoning in terms of exact concepts is carried out on lines of classical logic while reasoning with and about general concepts is approximative but degrees of approximation can be found from data. Several attempts has been made to built formal deductive systems for rough logic [81].

Probabilistic logic

A number of logics for approximative reasoning employ probabilistic semantics [1], [89]. It is based on evaluating evidence based probabilities of inferences $p \Rightarrow q$. In these evaluations often one applies *Bayesian reasoning*: in many applications e.g. in medicine it is relatively easy to establish probabilities $P(q \Rightarrow p)$, $P(p), P(q)$; from those, $P(p \Rightarrow q)$ is found via the Bayes formula as $P(q \Rightarrow p)P(p)$. Complex inferences may be carried out in semantic networks known as *Bayesian belief networks* i.e. graphical representations of causal relations in a domain. Having joint probability table $P(U)$ where $U = \{A_1, ..., A_n\}$ one can calculate $P(A_i)$ or $P(A_i \mid E)$ where E is an evidence. However the size of $P(U)$ grows exponentially with the number of variables. Therefore we look for compact representation of $P(U)$ from which $P(U)$ can be calculated if needed. Bayesian network over U is such a representation. If the conditional independence's in the Bayesian network hold for U then $P(U)$ can be calculated from the conditions specified in the network.

Inductive Reasoning about Knowledge

Inductive reasoning about knowledge presented above aims at evaluating the degree of assurance in validity of inferences $p \vdash q$. In this process, the semantics of inference is established relative to a parameterized family of connectives. The next step consists in finding approximations to connectives making inferences conforming with reality. Tasks of finding those approximations are often local, From these, elementary inference schemes are built and mechanisms for composing these schemes are found. Inference schemes are often desirable to be distributed (autonomous or hierarchical) due to high complexity of inference problems. On these schemes autoepistemic modalities may be superposed permitting the system to evaluate its knowledge and express metastatements about it. We include an example on constructing such schemes.

Localization (mereological logics)

Łukasiewicz measure μ_{L} of concept closeness may be generalized to a notion of rough inclusion [92] i.e. a predicate $\mu(X, Y)$ which for concepts X, Y returns the value of degree in which X is *a part of* Y. This predicate allows for approximate reasoning based on the notion of a partial containment: the inference $p \Rightarrow q$ is true in the degree $\mu([p], [q])$ where $[p]$ is the extension of the statement p. It seems that the understanding of μ should be local i.e. each intelligent agent should have its own collection of specific forms of rough inclusion and apply them locally again i.e. in distinct data substructures different rough inclusions may be valid. This idea involves necessity of a calculus for fusion of local rough inclusions at any agent as well as for propagation of rough inclusions among agents.

In search of a proper framework we turn to mereological theory of St. Leś - niewski [61]. This set theory has as the primitive notion the notion of a (*proper*) *part predicate*: $XpartY$ is required to be irreflexive and transitive; the improper

part predicate *ipart* is defined as follows: $XipartY$ if $XpartY$ or $X = Y$. The notion of a set is relativized to non-void properties: for any such property P, an object X is a set of objects with the property P ($XsetP$) if and only if given any Y with $YipartX$ there exist objects Q, R with $QipartY, QipartR, RipartX, R$ having P. A universal set for P is a class of P; one requires the uniqueness of class for each P. In this theory X is said to be an element of Y if for some property P, X has P and Y is class of P. The notion of a subset is extensional: X is a subset of Y if for each Z: Z element of X implies Z element of Y. It turns out that to be an element, to be a subset, and to be an improper part are all equivalent.

One may want to define a *rough inclusion* μ in such a way that a hierarchy of partial relations created by it contains a hierarchy of objects according to a part relation in the sense of Leśniewski. A set of postulates to this end may be as follows [92]: (1) $\mu(x, x) = \omega$; (2) $\mu(x, y) = \omega \Rightarrow (\mu(z, y) \geq \mu(z, x))$; (3) $(\mu(x, y) = \omega \wedge \mu(y, x) = \omega) \Rightarrow (\mu(x, z) \geq \mu(y, z))$; (4) $(\mu(z, x) = \omega) \Rightarrow \exists w : (\mu(w, z) = \omega \wedge \mu(w, y) = \omega) \Rightarrow (\mu(x, y) = \omega)$. An additional set of axiom schemes may guarantee class uniqueness. A model is a set M along with a 2 -ary function F from M into a lattice L (let us set $L = [0, 1]$) and the constant ω interpreted as $maxL$ (in our case, 1); in this interpretation $\mu(x, y)$ becomes $F(v(x), v(y))$ i.e. degree in which the element $v(x)$ of D is a part of $v(y)$. Letting in D: $v(x)partv(y)$ if $F(v(x), v(y)) = 1$ gives a relation of part in the sense of Leśniewski. Hence, a model for μ is also a model for fuzzy inference.

Reasoning on basis of μ requires a calculus for fusion of local inclusions at any agent reasoning about the world as well as a calculus for propagation of uncertainty among agents in order to ensure the correctness of inference over a scheme of agents. Logical calculus for an agent ag may be defined over a predicate calculus $L(ag)$: elementary formulas are of the form $\langle ag, \Phi(ag), \epsilon(ag)\rangle$ where $\Phi(ag)$ is a predicate of $L(ag)$ and $\epsilon(ag) \in [0, 1]$. Fusion of local calculi is achieved by selecting a subset $St(ag) \subseteq D(ag)$ of *standards*: to each standard st a local rough inclusion μ_{st} is attached. Then we say that an object $x \in D(ag)$ satisfies $\langle ag, \Phi(ag), \epsilon(ag)\rangle$ if and only if there exists a standard st such that : (i) st satisfies $\Phi(ag)$ (ii) $\mu_{st}(x, st) \geq 1 - \epsilon(ag)$. Propagation of uncertainty is achieved by means of mereological connectives f extracted from data; their role is to state that when children of ag submit objects satisfying their formulas in degrees say $\epsilon_1, ..., \epsilon_n$ then the object composed them satisfies a formula at ag in degree at least $f(\epsilon_1, ..., \epsilon_n)$ [92]. Schemes of agents may be composed to create inference engines for approximate proofs of complex formulas from atomic ones.

As with fuzzy logic and bayesian reasoning, application of this logic is analytical: first, the necessary ingredients have to be learned on training data and then tested on test data.

KDD as a logical process

Data mining can be described as searching from data for relevant structures (semantical models of logics) and their primitive properties i.e. *patterns*. These relevant constructs are used to discover knowledge. The process of knowledge

discovery can be treated as a kind of inference process (classical, commonsense etc.) based on the constructs found in data mining stage leading to efficient solutions of tasks like classification, prediction, etc. The solutions provide us with approximate descriptions of concepts of interest having satisfactory quality. The inference process has its own logic: in some cases it may be based on classical logic but in many cases, due to uncertainty, the logic of inference should be extracted from data as schemes for approximate reasoning [92], [132]. In this latter case a very important issue is knowledge granulation and reasoning with information granules [133, 134], [97] making feasible the reasoning process in case of complex problems like spatial reasoning [118] where the perception mechanisms play important role. Finally, let us mention that the inference process should be dynamically adapted to changing data. This causes the necessity to develop adaptive reasoning strategies tuning parameters of logical models (structures) and formulas to induce the optimal (sub-optimal) concept approximations.

KDD faces currently problems related to cognitive and information aspects of perception and reasoning [104]. This certainly stimulates investigations on foundations of logic towards the revision and redefining of its traditional notions [12]. For example, the notion of a proof seems to evolve towards the notion of an approximate scheme of reasoning (extracted from data) due to uncertainty or complexity of search in possible proof space.

Acknowledgement. This work has been supported by the grant No. 8T11C-02417 from the State Committee for Scientific Research (KBN) of the Republic of Poland and by the ESPRIT-CRIT 2 project #20288. Andrzej Skowron has also been partially supported by a grant of the Wallenberg Foundation and by a grant from the State Committee for Scientific Research (KBN) of the Republic of Poland.

References

1. Adams, E.W.: The Logic of Conditionals, An Application of Probability to Deductive Logic. D. Reidel Publishing Company, Dordrecht, 1975.
2. T. Agotnes, J. Komorowski, T. Loken : Taming large rule models in rough set approaches. Proceedings of the 3rd European Conference of Principles and Practice of Knowledge Discovery in Databases, September 15-18, 1999, Prague, Czech Republic, Lecture Notes in Artificial Intelligence **1704**, Springer-Verlag, Berlin, 1999, pp. 193-203.
3. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings ACM SIGMOD Conference on Management of Data, Washington, 1993, pp. 207-213.
4. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. Verkano(1996): Fast discovery of association rules. Fayyad U.M., Piatetsky-Shapiro G., Smyth P., Uthurusamy R. (Eds.): Advances in Knowledge Discovery and Data Mining, The AAAI Press/The MIT Press 1996, pp. 307-328.
5. Anderson, A.R., Belnap. N.D.: Entailment: The Logic of Relevance and Necessity. Princeton University Press, Vol.1, 1975.
6. Anderson, A.R., Belnap. N.D., Dunn, J.M: Entailment: Vol.2. Oxford University Press, 1992.

7. Anthony, M., Biggs, N.: Computational Learning Theory. Cambridge University Press, Cambridge, 1992.
8. Aquist, L.: Deontic logic.In: [31], pp. 605-714.
9. Bazan, J.G., Nguyen,Son Hung, Nguyen, Tuan Trung, Skowron, A., Stepaniuk, J.: Decision rules synthesis for object classification. In: E. Orlowska (ed.), Incomplete Information: Rough Set Analysis, Physica - Verlag, Heidelberg, 1998, pp. 23–57.
10. J. G. Bazan: A comparison of dynamic and non-dynamic rough set methods for extracting laws from decision system. In: [94], 1998, pp. 321–365.
11. van Bentham, J.: Temporal logic. In: [34],1995, pp. 241-350.
12. van Bentham, J.: Logic after the Golden Age. IILC Magazine, December 1999, Institute for Logic, Language and Computation, Amsterdam University, p. 12.
13. F. M. Brown: Boolean Reasoning. Kluwer Academic Publishers, Dordrecht, 1990.
14. G. Cattaneo: Abstract approximation spaces for rough theories. In: Polkowski and Skowron [94], 1998, pp. 59–98.
15. M. R. Chmielewski, J. W. Grzymala-Busse: Global discretization of attributes as preprocessing for machine learning. In: Proceedings of the Third International Workshop on Rough Sets and Soft Computing (RSSC'94),San Jose State University, San Jose, California, USA, November 10–12, 1994, pp. 294–301.
16. J. Cios, W. Pedrycz, and R.W. Swiniarski: Data Mining in Knowledge Discovery. Kluwer Academic Publishers, Dordrecht, 1998.
17. E. F. Codd: A relational model for large shared data banks. Comm. ACM. Vol.13, No 6, 1970, pp. 377-382.
18. A. Czyżewski: Soft processing of audio signals. In: Polkowski and Skowron [95], 1998, pp. 147–165.
19. Davis, M.: Computability and Unsolvability. Mc Graw-Hill, New York, 1958.
20. J. Deogun, V. Raghavan, A. Sarkar, and H. Sever: Data mining: Trends in research and development. In: Lin and Cercone [59], 1997, pp. 9–45.
21. Dougherty J., Kohavi R., and Sahami M.: Supervised and unsupervised discretization of continuous features. In: Proceedings of the Twelfth International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1995..
22. Dubois, D., Prade, H.: Belief Change. Vol. 3 in: Gabbay, D.M., Smets Ph. (eds.), Handbook of Defeasible Reasoning and Uncertainty Management Systems. Kluwer Academic Publishers, Dordrecht, 1998.
23. I. Duentsch, G. Gediga: Statistical evaluation of rough set dependency analysis. International Journal of Human-Computer Studies **46**, 1997, pp. 589-604.
24. I. Duentsch, G. Gediga: Rough set data analysis. In: Encyclopedia of Computer Science and Technology, Marcel Dekker (to appear).
25. M. J. Egenhofer, R. G. Golledge (eds.): Spatial and Temporal Reasoning in Geographic Information Systems. Oxford University Press, Oxford, 1997.
26. Eisinger, M., Ohlbach, H. J.: Deduction systems based on resolution. In: [32], pp. 183-271.
27. Fagin, R., Halpern. J.Y., Moses, Y., and Vardi, M.Y.: Reasoning about Knowledge. MIT Press, Cambridge MA, 1995.
28. Fayyad, U., Piatetsky-Shapiro, G. (eds.): Advances in Knowledge Discovery and Data Mining. MIT and AAAI Press, Cambridge MA, 1996.
29. Fitting, M.: Basic modal logic. In: [33], pp. 368-438.
30. J. Friedman, R. Kohavi, and Y. Yun: Lazy Decision Trees. In: Proc. of AAAI-96, 1996, pp. 717–724.
31. Gabbay, D., Guenthner, F. (eds.): Handbook of Philosophical Logic Vol.2. Kluwer Academic Publishers, Dordrecht, 1994.

32. Gabbay, D.M., Hogger, C.J., and Robinson, J.A. (eds.): Handbook of Logic in Artificial Intelligence and Logic Programming Vol.1. Oxford University Press, New York, 1993.
33. Gabbay, D.M., Hogger, C.J., and Robinson, J.A. (eds.): Handbook of Logic in Artificial Intelligence and Logic Programming vol.3. Oxford University Press, New York, 1993.
34. Gabbay, D.M., Hogger, C.J., and Robinson, J.A. (eds.): Handbook of Logic in Artificial Intelligence and Logic Programming Vol.4. Oxford University Press, New York, 1995.
35. Garey, M.R., Johnson, D.S.: Computers and Intractability. A Guide to the Theory of NP-completeness. W.H. Freeman and Company, New York, 1979.
36. Gärdenfors, P., Rott, H.: Belief revision. In: [34], 1995, pp. 35-132.
37. Geffner, H.: Default Reasoning: Causal and Conditional theories. MIT Press, Cambridge, MA, 1992.
38. Gentzen, G.: Untersuchungen über das logische Schliessen, Math. Zeitschrift **39**, 1934, pp. 176-210, 405-431.
39. C. Glymour, D. Madigan, D. Pregibon, and P. Smyth: Statistical themes and lessons for data mining. Data Mining and Knowledge Discovery **1**, 1996, pp. 25-42.
40. Gonzalez. A.J., Dankel, D.D.: The Engineering of Knowledge Based Systems: Theory and Practice. Prentice Hall, Englewood Cliffs, NJ, 1993.
41. Gödel, K.: die Vollständigkeit der Axiome des Logischen Funktionenkalküls, Monatshefte für Mathematik und Physik **37**, 1930, pp. 349-360.
42. Gödel, K.: Über formal unetscheidbare Sätze der Principia Mathematica und Verwandtersysteme I, Monatshefte für Mathematik und Physik **38**, 1931, pp. 173-198.
43. S. Greco, B. Matarazzo, and R. Słowinski: Rough Approximation of a Preference Relation in a Pairwise Comparison Table. In: Polkowski and Skowron [95], 1998, pp. 13–36.
44. J.W. Grzymała–Busse: Applications of the rule induction system LERS. In: Polkowski and Skowron [94], 1998, pp. 366–375.
45. Halpern, J.Y.: Reasoning about knowledge: a survey. In: [34], 1995, pp. 1–34.
46. Harel D.: Dynamic logic. In: [31], 1994, pp. 497–604.
47. Hintikka, J.: Knowledge and Belief. Cornell University Press, Ithaca, N.Y., 1962.
48. Hodges, W.: Logical features of Horn clauses. In: [32], 449-503.
49. P. J. Huber: Robust statistics. Wiley, New York, 1981.
50. Hughes, G.E., Creswell, M.J.: An Introduction to Modal Logic. Methuen, London, 1968.
51. J. Komorowski, J. Żytkow (Eds.): The First European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'97). June 25–27, Trondheim, Norway. Lecture Notes in Artificial Intelligence **1263**, Springer-Verlag, Berlin, 1997, pp. 1–396.
52. J. Komorowski, Z. Pawlak, L. Polkowski, and A. Skowron Rough sets: A tutorial. In: S.K. Pal and A. Skowron (eds.), Rough fuzzy hybridization: A new trend in decision–making. Springer-Verlag, Singapore, 1997, pp. 3-98.
53. Konolige, K.: Autoepistemic logic. In: [33], 1994, 217-295.
54. W. Kowalczyk: Rough data modelling, A new technique for analyzing data. In: Polkowski and Skowron [94], 1998, pp. 400–421.
55. K. Krawiec, R. Słowiński, and D. Vanderpooten: Learning decision rules from similarity based rough approximations. In: Polkowski and Skowron [95], 1998, pp. 37–54.

56. Kruse, R., Gebhardt, J., and Klawonn, F.: Foundations of Fuzzy Systems. J. Wiley, New York, 1994.
57. M. Kryszkiewicz: Generation of rules from incomplete information systems. In: Komorowski and Żytkow [51], 1997, pp. 156–166.
58. Lifschitz, V.: Circumscription. In: [33], 297-352.
59. T. Y. Lin, N. Cercone (Eds.): Rough Sets and Data Mining. Analysis of Imprecise Data. Kluwer Academic Publishers, Boston, 1997.
60. T.Y. Lin: Granular computing on binary relations I, II. In: Polkowski and Skowron [94], 1998, pp. 107–140.
61. Leśniewski, S.: On the foundations of mathematics. In: Surma, S., Srzednicki,J.T., Barnett, D.I., Rickey, F.V. (eds), Stanislaw Leśniewski. Collected Works. Kluwer Academic Publishers, Dordrecht (1992), pp. 174-382.
62. Lloyd, J. W.: Foundations of Logic Programming. Springer- Verlag, Berlin, 1984.
63. Łukasiewicz, J.: Die logischen Grundlagen der Wahrscheinchkeitsrechnung. Kraków, 1913.
64. Łukasiewicz, J.: Philosophische Bemerkungen zu mehrwertigen Systemen des Aussagenkalküls. Comptes rendus de la Société des Sciences et des Lettres de Varsovie **23**, 1930, pp. 57-77.
65. Makinson, D.: General patterns in non-monotonic reasoning. In: [33], 35-110.
66. Mendelson, E.: Introduction to Mathematical Logic. Van Nostrand, New York, 1960.
67. Michalski, R.: Inferential theory of learning as a conceptual basis for multistrategy. Machine Learning **11**, 1993, pp. 111-151.
68. Michalski R., Tecuci G.: Machine Learning. A Multistrategy Approach Vol.4. Morgan Kaufmann, San Francisco, 1994.
69. D. Michie, D.J. Spiegelhalter, C.C. Taylor (Eds.): Machine Learning, Neural and Statistical Classification. Ellis Horwood, New York, 1994.
70. T.M. Mitchell: Machine Learning. Mc Graw-Hill, Portland, 1997.
71. T. Mollestad and J. Komorowski: A Rough Set Framework for Propositional Default Rules Data Mining. In: S.K. Pal and A. Skowron (Eds.): Rough – Fuzzy Hybridization: New Trend in Decision Making. Springer–Verlag, Singapore, 1999.
72. Muggleton, S.: Foundations of Inductive Logic Programming. Prentice Hall, Englewood Cliffs, 1995.
73. Nadler M., Smith E.P.: Pattern Recognition Engineering. Wiley, New York, 1993.
74. H. S. Nguyen: Discretization of Real Value Attributes, Boolean Reasoning Approach. Ph.D. Dissertation, Warsaw University, 1997, pp. 1–90.
75. H. S. Nguyen: Efficient SQL-learning method for data mining in large data bases. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99), 1999, pp. 806–811.
76. S. H. Nguyen: Data Regularity Analysis and Applications in Data Mining. Ph.D. Dissertation, Warsaw University, 1999.
77. H.S. Nguyen, S.H. Nguyen: Pattern extraction from data. Fundamenta Informaticae **34**, 1998, pp. 129–144.
78. H.S. Nguyen, S.H. Nguyen: Rough sets and association rule generation. Fundamenta Informaticae **40/4** (in print).
79. S. H. Nguyen, A. Skowron, and P. Synak: Discovery of data patterns with applications to decomposition and classification problems. In: Polkowski and Skowron [95], 1998, pp. 55–97.
80. Nute, D.: Conditional logic. In: [31], 387-440.

81. E. Orłowska (ed.): Incomplete Information: Rough Set Analysis. Physica–Verlag, Heidelberg, 1998.
82. S. K. Pal, A. Skowron (1999), Rough–fuzzy Hybridization: A New Trend in Decision Making. Springer–Verlag, Singapore, 1999.
83. G. Paun, L. Polkowski, and A. Skowron: Parallel communicating grammar systems with negotiations. Fundamenta Informaticae **28/3-4**, 1996, pp. 315–330.
84. Z. Pawlak: Information systems – theoretical foundations. Information Systems **6**, 1981, pp. 205–218.
85. Z. Pawlak: Rough sets. International Journal of Computer and Information Sciences **11**, 1982, pp. 341–356.
86. Z. Pawlak: Rough Sets: Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht, 1992.
87. Z. Pawlak, Z. Ras, (Eds.): Proceedings: Ninth International Symposium on Methodologies for Intelligent Systems (ISMIS'96),Springer–Verlag, Berlin, 1996.
88. Z. Pawlak, A. Skowron: Rough set rudiments. Bulletin of the International Rough Set Society **3/4**, 1999, pp. 181-185.
89. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco, 1988.
90. Z. Piasta, A. Lenarcik: Rule induction with probabilistic rough classifiers. Machine Learning (to appear).
91. L. Polkowski: On synthesis of constructs for spatial reasoning via rough mereology. Fundamenta Informaticae (to appear).
92. L. Polkowski, A. Skowron: Rough mereology: A new paradigm for approximate reasoning. International Journal of Approximate Reasoning **15/4**,1994, pp. 333–365.
93. L. Polkowski, A. Skowron: Rough sets: A perspective. In: Polkowski and Skowron [94], 1998, pp. 31–58.
94. L. Polkowski, A. Skowron (Eds.): Rough Sets in Knowledge Discovery 1: Methodology and Applications. Physica-Verlag, Heidelberg, 1998.
95. L. Polkowski, A. Skowron (Eds.: Rough Sets in Knowledge Discovery 2: Applications, Case Studies and Software Systems. Physica-Verlag, Heidelberg, 1998.
96. L. Polkowski, A. Skowron (Eds.): Proc. First International Conference on Rough Sets and Soft Computing, RSCTC'98, Warszawa, Poland, LNAI **1424**, Springer-Verlag, Berlin, 1998.
97. L. Polkowski, A. Skowron: Towards adaptive calculus of granules. In: [135], **1**, 1999, pp. 201–227.
98. Poole, D.: Default logic. In: [33], 189-216.
99. Prawitz, D.: Natural deduction, a proof theoretic study. Stockholm Studies in Philosophy Vol.3, Almquist & Wiksell, Stockholm, 1965.
100. J.R. Quinlan: C4.5. Programs for machine learning. Morgan Kaufmann, San Mateo, CA, 1993.
101. Z.W. Ras: Cooperative knowledge–based systems. Journal of the Intelligent Automation Soft Computing **2/2** (special issue edited by T.Y. Lin), 1996, pp. 193–202.
102. Rescher, N.: Many-valued Logics. Mc Graw Hill, New York, 1969.
103. J. J. Rissanen: Modeling by Shortest Data Description. Automatica **14**, 1978, pp. 465-471.
104. Roddick J.F., Spiliopoulou M.: A Bibliography of Temporal, Spatial, and Temporal Data Mining Research. Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining **1/1**, 1999, pp. 34-38.

105. Rosser, J.B., Turquette, A.R.: Many-valued Logics. North-Holland, Amsterdam, 1952.
106. Russel, S.J., Norvig, P.: Artificial Intelligence. A Modern Approach. Prentice Hall, Englewood Cliffs, 1995.
107. Sandewall, E., Shoham, Y.: Non-monotonic temporal reasoning. In: [34], 439-498.
108. B. Selman, H. Kautz and D. McAllester. *Ten challenges in propositional reasoning and search.* Proc. IJCAI'97, Japan.
109. Skowron A.: Synthesis of adaptive decision systems from experimental data. In: A. Aamodt, J. Komorowski (eds): Proc. of the Fifth Scandinavian Conference on Artificial Intelligence (SCAI'95), May 1995, Trondheim, Norway. IOS Press, Amsterdam, 1995, pp. 220–238.
110. A. Skowron, H.S. Nguyen: Boolean reasoning scheme with some applications in data mining. Proceedings of the 3-rd European Conference on Principles and Practice of Knowledge Discovery in Databases, September 1999, Prague, Czech Republic. Lecture Notes in Computer Science **1704**, 1999, pp. 107–115.
111. A. Skowron, C. Rauszer: The Discernibility Matrices and Functions in Information Systems. In: Słowiński [115], 1992, pp. 331–362.
112. A. Skowron, J. Stepaniuk: Tolerance Approximation Spaces. Fundamenta Informaticae **27**, 1996, pp. 245–253.
113. A. Skowron, J. Stepaniuk, and S. Tsumoto: Information Granules for Spatial Reasoning. Bulletin of the International Rough Set Society **3/4**, 1999, pp. 147-154.
114. D. Ślęzak: Approximate reducts in decision tables. In: Proceedings of the Sixth International Conference, Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'96) vol. 3, July 1–5, Granada, Spain, 1996, pp. 1159–1164.
115. R. Słowiński, (Ed.): Intelligent Decision Support – Handbook of Applications and Advances of the Rough Sets Theory. Dordrecht, Kluwer Academic Publishers, 1992.
116. R. Słowiński, D. Vanderpooten: Similarity relation as a basis for rough approximations. In: P. Wang (Ed.): Advances in Machine Intelligence & Soft Computing. Bookwrights, Raleigh NC, 1997, pp. 17–33.
117. R. Słowiński, D. Vanderpooten: A generalized definition of rough approximations based on similarity. IEEE Trans. on Data and Knowledge Engineering (to appear).
118. `http:agora.leeds.ac.ukspacenet.html`
119. Stalnaker, R.: A theory of conditionals. In: N. Rescher (ed.): Studies in Logical Theory. Blackwell, Oxford, 1968.
120. M. Szczuka: Symbolic and neural network methods for classifiers construction. Ph.D. Dissertation, Warsaw University, 1999.
121. Tarski, A.: On the concept of logical consequence. In: Logic, Semantics, Metamathematics. Oxford University Press, Oxford, 1956.
122. Tarski, A.: Der Wahrheitsbegriff in den Formalisierten Sprachen, Studia Philosophica **1**, 1936, pp. 261-405.
123. Troelstra, A. S.: Aspects of constructive mathematics. In: Barwise, J. (ed.): Handbook of Mathematical Logic. North Holland, Amsterdam, 1977, pp. 973-1052.
124. Tsang, E.: Foundations of Constraint Satisfaction. Academic Press, London 1993.
125. S. Tsumoto: Modelling diagnostic rules based on rough sets. In: Polkowski and Skowron [96], 1998, pp. 475–482.
126. J. D. Ullman, J. Widom: A First Course in Database Systems. Prentice–Hall, Inc., Englewood Cliffs, 1997.

127. Urquhart, A.: Many-valued logic. In: [33], 71-116.
128. A. Wasilewska, L. Vigneron: Rough algebras and automated deduction. In: Polkowski and Skowron [94], 1998, pp. 261–275.
129. Wille, R.: Formale Begriffsanalyse: Mathematische Grundlagen. Springer-Verlag, Berlin, 1996.
130. Von Wright, G.H.: An Essay in Modal Logic. North Holland, Amsterdam, 1951.
131. Zadeh, L.A.: Fuzzy sets. Information and Control **8**, 1965, pp. 333-353.
132. Zadeh, L.A.: A theory of approximate reasoning. In: Hayes, J.E., Michie, D., Mikulich, L.C. (eds.): Machine Intelligence Vol.9. J. Wiley, New York, 1979, pp. 149-194.
133. Zadeh, L.A.: Fuzzy logic = computing with words. IEEE Trans. on Fuzzy Systems **4**, 1996, pp. 103-111.
134. Zadeh, L.A: Toward a theory of fuzzy information granulation and its certainty in human reasoning and fuzzy logic. Fuzzy Sets and Systems **90**, 1997, pp. 111-127.
135. Zadeh, L.A., Kacprzyk, J. (eds.): Computing with Words in Information/Intelligent Systems Vol. 1-2. Physica-Verlag, Heidelberg, 1999.
136. M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li: New parallel algorithms for fast discovery of association rules. In: Data Mining and Knowledge Discovery : An International Journal (special issue on Scalable High-Performance Computing for KDD) **1/4**, 1997, pp. 343–373.
137. W. Ziarko: Variable Precision Rough Set Model. J. of Computer and System Sciences **46**, 1993, pp. 39–59.
138. W. Ziarko (ed.): Rough Sets, Fuzzy Sets and Knowledge Discovery (RSKD'93). Workshops in Computing, Springer–Verlag & British Computer Society, London, Berlin, 1994.
139. W. Ziarko: Rough sets as a methodology for data mining. In: Polkowski and Skowron [95], 1998, pp. 554–576.