Data with Missing Attribute Values: Generalization of Indiscernibility Relation and Rule Induction

Jerzy W. Grzymala-Busse[1,2]

[1] Department of Electrical Engineering and Computer Science, University of Kansas
Lawrence, KS 66045, USA
[2] Institute of Computer Science, Polish Academy of Sciences, 01-237 Warsaw, Poland
Jerzy@ku.edu
http://lightning.eecs.ku.edu/index.html

Abstract. Data sets, described by decision tables, are incomplete when for some cases (examples, objects) the corresponding attribute values are missing, e.g., are lost or represent "do not care" conditions. This paper shows an extremely useful technique to work with incomplete decision tables using a block of an attribute-value pair. Incomplete decision tables are described by characteristic relations in the same way complete decision tables are described by indiscernibility relations. These characteristic relations are conveniently determined by blocks of attribute-value pairs. Three different kinds of lower and upper approximations for incomplete decision tables may be easily computed from characteristic relations. All three definitions are reduced to the same definition of the indiscernibility relation when the decision table is complete. This paper shows how to induce certain and possible rules for incomplete decision tables using MLEM2, an outgrow of the rule induction algorithm LEM2, again, using blocks of attribute-value pairs. Additionally, the MLEM2 may induce rules from incomplete decision tables with numerical attributes as well.

1 Introduction

We will assume that data sets are presented as decision tables. In such a table columns are labeled by variables and rows by case names. In the simplest case such case names, also called cases, are numbers. Variables are categorized as either independent, also called attributes, or dependent, called decisions. Usually only one decision is given in a decision table. The set of all cases that correspond to the same decision value is called a concept (or a class).

In most articles on rough set theory it is assumed that for all variables and all cases the corresponding values are specified. For such tables the indiscernibility relation, one of the most fundamental ideas of rough set theory, describes cases that can be distinguished from other cases.

However, in many real-life applications, data sets have missing attribute values, or, in other words, the corresponding decision tables are incompletely spec-

J.F. Peters et al. (Eds.): Transactions on Rough Sets I, LNCS 3100, pp. 78–95, 2004.

ified. For simplicity, incompletely specified decision tables will be called incomplete decision tables.

In this paper we will assume that there are two reasons for decision tables to be incomplete. The first reason is that an attribute value, for a specific case, is lost. For example, originally the attribute value was known, however, due to a variety of reasons, currently the value is not recorded. Maybe it was recorded but is erased. The second possibility is that an attribute value was not relevant – the case was decided to be a member of some concept, i.e., was classified, or diagnosed, in spite of the fact that some attribute values were not known. For example, it was feasible to diagnose a patient regardless of the fact that some test results were not taken (here attributes correspond to tests, so attribute values are test results). Since such missing attribute values do not matter for the final outcome, we will call them "do not care" conditions. The main objective of this paper is to study incomplete decision tables, i.e., incomplete data sets, or, yet in different words, data sets with missing attribute values. We will assume that in the same decision table some attribute values may be lost and some may be "do not care" conditions. The first paper dealing with such decision tables was [6].

For such incomplete decision tables there are two special cases: in the first case, all missing attribute values are lost, in the second case, all missing attribute values are "do not care" conditions. Incomplete decision tables in which all attribute values are lost, from the viewpoint of rough set theory, were studied for the first time in [8], where two algorithms for rule induction, modified to handle lost attribute values, were presented. This approach was studied later in [13–15], where the indiscernibility relation was generalized to describe such incomplete decision tables.

On the other hand, incomplete decision tables in which all missing attribute values are "do not care" conditions, from the view point of rough set theory, were studied for the first time in [3], where a method for rule induction was introduced in which each missing attribute value was replaced by all values from the domain of the attribute. Originally such values were replaced by all values from the entire domain of the attribute, later, by attribute values restricted to the same concept to which a case with a missing attribute value belongs. Such incomplete decision tables, with all missing attribute values being "do not care conditions", were extensively studied in [9], [10], including extending the idea of the indiscernibility relation to describe such incomplete decision tables.

In general, incomplete decision tables are described by characteristic relations, in a similar way as complete decision tables are described by indiscernibility relations [6].

In rough set theory, one of the basic notions is the idea of lower and upper approximations. For complete decision tables, once the indiscernibility relation is fixed and the concept (a set of cases) is given, the lower and upper approximations are unique.

For incomplete decision tables, for a given characteristic relation and concept, there are three different possibilities to define lower and upper approximations,

called singleton, subset, and concept approximations [6]. Singleton lower and upper approximations were studied in [9], [10], [13–15]. Note that similar three definitions of lower and upper approximations, though not for incomplete decision tables, were studied in [16–18]. In this paper we further discuss applications to data mining of all three kinds of approximations: singleton, subset and concept. As it was observed in [6], singleton lower and upper approximations are not applicable in data mining.

The next topic of this paper is demonstrating how certain and possible rules may be computed from incomplete decision tables. An extension of the well-known LEM2 algorithm [1], [4], MLEM2, was introduced in [5]. Originally, MLEM2 induced certain rules from incomplete decision tables with missing attribute values interpreted as lost and with numerical attributes. Using the idea of lower and upper approximations for incomplete decision tables, MLEM2 was further extended to induce both certain and possible rules from a decision table with some missing attribute values being lost and some missing attribute values being "do not care" conditions, while some attributes may be numerical.

2 Blocks of Attribute-Value Pairs and Characteristic Relations

Let us reiterate that our basic assumption is that the input data sets are presented in the form of a *decision table*. An example of a decision table is shown in Table 1.

Table 1. A complete decision table

	Attributes			Decision
Case	Temperature	Headache	Nausea	Flu
1	high	yes	no	yes
2	very_high	yes	yes	yes
3	high	no	no	no
4	high	yes	yes	yes
5	high	yes	yes	no
6	normal	yes	no	no
7	normal	no	yes	no
8	normal	yes	no	yes

Rows of the decision table represent *cases*, while columns are labeled by *variables*. The set of all cases will be denoted by U. In Table 1, $U = \{1, 2, ..., 8\}$. Independent variables are called *attributes* and a dependent variable is called a *decision* and is denoted by d. The set of all attributes will be denoted by A. In Table 1, $A = \{Temperature, Headache, Nausea\}$. Any decision table defines a function ρ that maps the direct product of U and A into the set of all values. For example, in Table 1, $\rho(1, Temperature) = high$. Function ρ describing Table 1 is completely specified (total). A decision table with completely specified function ρ will be called *completely specified*, or, for the sake of simplicity, *complete*.

Rough set theory [11], [12] is based on the idea of an indiscernibility relation, defined for complete decision tables. Let B be a nonempty subset of the set A of all attributes. The indiscernibility relation $IND(B)$ is a relation on U defined for $x, y \in U$ as follows

$$(x, y) \in IND(B) \text{ } if \text{ } and \text{ } only \text{ } if \text{ } \rho(x, a) = \rho(y, a) \text{ } for \text{ } all \text{ } a \in B.$$

The indiscernibility relation $IND(B)$ is an equivalence relation. Equivalence classes of $IND(B)$ are called elementary sets of B and are denoted by $[x]_B$. For example, for Table 1, elementary sets of $IND(A)$ are {1}, {2}, {3}, {4, 5}, {6, 8}, {7}. The indiscernibility relation $IND(B)$ may be computed using the idea of blocks of attribute-value pairs. Let a be an attribute, i.e., $a \in A$ and let v be a value of a for some case. For complete decision tables if $t = (a, v)$ is an attribute-value pair then a block of t, denoted $[t]$, is a set of all cases from U that for attribute a have value v. For Table 1,

[(Temperature, high)] = {1, 3, 4, 5},
[(Temperature, very_high)] = {2},
[(Temperature, normal)] = {6, 7, 8},
[(Headache, yes)] = {1, 2, 4, 5, 6, 8},
[(Headache, no)] = {3, 7},
[(Nausea, no)] = {1, 3, 6},
[(Nausea, yes)] = {2, 4, 5, 7}.

The indiscernibility relation $IND(B)$ is known when all elementary blocks of IND(B) are known. Such elementary blocks of B are intersections of the corresponding attribute-value pairs, i.e., for any case $x \in U$,

$$[x]_B = \cap\{[(a, v)] | a \in B, \rho(x, a) = v\}.$$

We will illustrate the idea how to compute elementary sets of B for Table 1 and $B = A$.

$[1]_A = [(Temperature, high)] \cap [(Headache, yes)] \cap [(Nausea, no)] = \{1\}$,
$[2]_A = [(Temperature, very\_high)] \cap [(Headache, yes)] \cap [(Nausea, yes)] = \{2\}$,
$[3]_A = [(Temperature, high)] \cap [(Headache, no)] \cap [(Nausea, no)] = \{3\}$,
$[4]_A = [5]_A = [(Temperature, high)] \cap [(Headache, yes)] \cap [(Nausea, yes)] = \{4, 5\}$,
$[6]_A = [8]_A = [(Temperature, normal)] \cap [(Headache, yes)] \cap [(Nausea, no] = \{6, 8\}$,
$[7]_A = [(Temperature, normal)] \cap [(Headache, no] \cap [(Nausea, yes)] = \{7\}$.

In practice, input data for data mining are frequently affected by missing attribute values. In other words, the corresponding function ρ is incompletely specified (partial). A decision table with an incompletely specified function ρ will be called *incompletely specified*, or *incomplete*.

For the rest of the paper we will assume that all decision values are specified, i.e., they are not missing. Also, we will assume that all missing attribute values are denoted either by "?" or by "*", lost values will be denoted by "?", "do not

Table 2. An incomplete decision table

	Attributes			Decision
Case	Temperature	Headache	Nausea	Flu
1	high	?	no	yes
2	very_high	yes	yes	yes
3	?	no	no	no
4	high	yes	yes	yes
5	high	?	yes	no
6	normal	yes	no	no
7	normal	no	yes	no
8	*	yes	*	yes

care" conditions will be denoted by "*". Additionally, we will assume that for each case at least one attribute value is specified.

Incomplete decision tables are described by characteristic relations instead of indiscernibility relations. Also, elementary blocks are replaced by characteristic sets. An example of an incomplete table is presented in Table 2.

For incomplete decision tables the definition of a block of an attribute-value pair must be modified. If for an attribute a there exists a case x such that $\rho(x, a) =?$, i.e., the corresponding value is lost, then the case x is not included in the block $[(a, v)]$ for any value v of attribute a. If for an attribute a there exists a case x such that the corresponding value is a "do not care" condition, i.e., $\rho(x, a) = *$, then the corresponding case x should be included in blocks $[(a, v)]$ for all values v of attribute a. This modification of the definition of the block of attribute-value pair is consistent with the interpretation of missing attribute values, lost and "do not care" condition. Thus, for Table 2

[(Temperature, high)] = {1, 4, 5, 8},
[(Temperature, very_high)] = {2, 8},
[(Temperature, normal)] = {6, 7, 8},
[(Headache, yes)] = {2, 4, 6, 8},
[(Headache, no)] = {3, 7},
[(Nausea, no)] = {1, 3, 6, 8},
[(Nausea, yes)] = {2, 4, 5, 7, 8}.

The *characteristic set* $K_B(x)$ is the intersection of blocks of attribute-value pairs (a, v) for all attributes a from B for which $\rho(x, a)$ is specified and $\rho(x, a) = v$. For Table 2 and $B = A$,

$K_A(1) = \{1, 4, 5, 8\} \cap \{1, 3, 6, 8\} = \{1, 8\}$,
$K_A(2) = \{2, 8\} \cap \{2, 4, 6, 8\} \cap \{2, 4, 5, 7, 8\} = \{2, 8\}$,
$K_A(3) = \{3, 7\} \cap \{1, 3, 6, 8\} = \{3\}$,
$K_A(4) = \{1, 4, 5, 8\} \cap \{2, 4, 6, 8\} \cap \{2, 4, 5, 7, 8\} = \{4, 8\}$,
$K_A(5) = \{1, 4, 5, 8\} \cap \{2, 4, 5, 7, 8\} = \{4, 5, 8\}$,
$K_A(6) = \{6, 7, 8\} \cap \{2, 4, 6, 8\} \cap \{1, 3, 6, 8\} = \{6, 8\}$,
$K_A(7) = \{6, 7, 8\} \cap \{3, 7\} \cap \{2, 4, 5, 7, 8\} = \{7\}$, and
$K_A(8) = \{2, 4, 6, 8\}$.

Characteristic set $K_B(x)$ may be interpreted as the smallest set of cases that are indistinguishable from x using all attributes from B, using a given interpretation of missing attribute values. Thus, $K_A(x)$ is the set of all cases that cannot be distinguished from x using all attributes.

The characteristic relation $R(B)$ is a relation on U defined for $x, y \in U$ as follows

$$(x, y) \in R(B) \text{ if and only if } y \in K_B(x).$$

The characteristic relation $R(B)$ is reflexive but – in general – does not need to be symmetric or transitive. Also, the characteristic relation $R(B)$ is known if we know characteristic sets $K(x)$ for all $x \in U$. In our example, $R(A) = \{(1, 1), (1, 8), (2, 2), (2, 8), (3, 3), (4, 4), (4, 8), (5, 4), (5, 5), (5, 8), (6, 6), (6, 8), (7, 7), (8, 2), (8, 4), (8, 6), (8, 8)\}$. The most convenient way is to define the characteristic relation through the characteristic sets. Nevertheless, the characteristic relation $R(B)$ may be defined independently of characteristic sets in the following way:

$$(x, y) \in R(B) \text{ if and only if } \rho(x, a) = \rho(y, a) \text{ or } \rho(x, a) = * \text{ or} \rho(y, a) = * \text{ for all } a \in B \text{ such that } \rho(x, a) \neq ?.$$

For decision tables, in which all missing attribute values are lost, a special characteristic relation was defined by J. Stefanowski and A. Tsoukias in [14], see also, e.g., [13], [15]. In this paper that characteristic relation will be denoted by $LV(B)$, where B is a nonempty subset of the set A of all attributes. For $x, y \in U$ characteristic relation $LV(B)$ is defined as follows:

$$(x, y) \in LV(B) \text{ if and only if } \rho(x, a) = r(y, a) \text{ for all } a \in B \text{ such that } \rho(x, a) \neq ?.$$

For any decision table in which all missing attribute values are lost, the characteristic relation $LV(B)$ is reflexive, but – in general – does not need to be symmetric or transitive.

For decision tables where all missing attribute values are "do not care" conditions a special characteristic relation, in this paper denoted by $DCC(B)$, was defined by M. Kryszkiewicz in [9], see also, e.g., [10]. For $x, y \in U$, the characteristic relation $DCC(B)$ is defined as follows:

$$(x, y) \in DCC(B) \text{ if and only if } \rho(x, a) = \rho(y, a) \text{ or} \rho(x, a) = * \text{ or } \rho(y, a) = * \text{ for all } a \in B.$$

Relation $DCC(B)$ is reflexive and symmetric but – in general – not transitive.

Obviously, characteristic relations $LV(B)$ and $DCC(B)$ are special cases of the characteristic relation $R(B)$. For a completely specified decision table, the characteristic relation $R(B)$ is reduced to $IND(B)$.

3 Lower and Upper Approximations

For completely specified decision tables lower and upper approximations are defined on the basis of the indiscernibility relation. Any finite union of elementary sets, associated with B, will be called a *B-definable set*. Let X be any subset of the set U of all cases. The set X is called a *concept* and is usually defined as the set of all cases defined by a specific value of the decision. In general, X is not a B-definable set. However, set X may be approximated by two B-definable sets, the first one is called a *B-lower approximation* of X, denoted by $\underline{B}X$ and defined as follows

$$\{x \in U | [x]_B \subseteq X\}.$$

The second set is called a *B-upper approximation* of X, denoted by $\overline{B}X$ and defined as follows

$$\{x \in U | [x]_B \cap X \neq \emptyset.$$

The above shown way of computing lower and upper approximations, by constructing these approximations from singletons x, will be called the *first method*. The B-lower approximation of X is the greatest B-definable set, contained in X. The B-upper approximation of X is the smallest B-definable set containing X.

As it was observed in [12], for complete decision tables we may use a *second method* to define the B-lower approximation of X, by the following formula

$$\cup\{[x]_B | x \in U, [x]_B \subseteq X\},$$

and the B-upper approximation of x may de defined, using the second method, by

$$\cup\{[x]_B | x \in U, [x]_B \cap X \neq \emptyset).$$

For incompletely specified decision tables lower and upper approximations may be defined in a few different ways. First, the definition of definability should be modified. Any finite union of characteristic sets of B is called a *B-definable* set. In this paper we suggest three different definitions of lower and upper approximations. Again, let X be a concept, let B be a subset of the set A of all attributes, and let $R(B)$ be the characteristic relation of the incomplete decision table with characteristic sets $K(x)$, where $x \in U$. Our first definition uses a similar idea as in the previous articles on incompletely specified decision tables [9], [10], [13], [14], [15], i.e., lower and upper approximations are sets of singletons from the universe U satisfying some properties. Thus, lower and upper approximations are defined by analogy with the above first method, by constructing both sets from singletons. We will call these approximations *singleton*. A singleton B-lower approximation of X is defined as follows:

$$\underline{B}X = \{x \in U | K_B(x) \subseteq X\}.$$

A singleton B-upper approximation of X is

$$\overline{B}X = \{x \in U | K_B(x) \cap X \neq \emptyset\}.$$

In our example of the decision table presented in Table 2 let us say that $B = A$. Then the singleton A-lower and A-upper approximations of the two concepts: {1, 2, 4, 8} and {3, 5, 6, 7} are:

$$\underline{A}\{1,2,4,8\} = \{1,2,4\},$$

$$\underline{A}\{3,5,6,7\} = \{3,7\},$$

$$\overline{A}\{1,2,4,8\} = \{1,2,4,5,6,8\},$$

$$\overline{A}\{3,5,6,7\} = \{3,5,6,7,8\}.$$

The second method of defining lower and upper approximations for complete decision tables uses another idea: lower and upper approximations are unions of elementary sets, subsets of U. Therefore we may define lower and upper approximations for incomplete decision tables by analogy with the second method, using characteristic sets instead of elementary sets. There are two ways to do this. Using the first way, a *subset* B-lower approximation of X is defined as follows:

$$\underline{B}X = \cup\{K_B(x)|x \in U, K_B(x) \subseteq X\}.$$

A *subset* B-upper approximation of X is

$$\overline{B}X = \cup\{K_B(x)|x \in U, K_B(x) \cap X \neq \emptyset\}.$$

Since any characteristic relation $R(B)$ is reflexive, for any concept X, singleton B-lower and B-upper approximations of X are subsets of the subset B-lower and B-upper approximations of X, respectively. For the same decision table, presented in Table 2, the subset A-lower and A-upper approximations are

$$\underline{A}\{1,2,4,8\} = \{1,2,4,8\},$$

$$\underline{A}\{3,5,6,7\} = \{3,7\},$$

$$\overline{A}\{1,2,4,8\} = \{1,2,4,5,6,8\},$$

$$\overline{A}\{3,5,6,7\} = \{2,3,4,5,6,7,8\}.$$

The second possibility is to modify the subset definition of lower and upper approximation by replacing the universe U from the subset definition by a concept X. A *concept* B-lower approximation of the concept X is defined as follows:

$$\underline{B}X = \cup\{K_B(x)|x \in X, K_B(x) \subseteq X\}.$$

Obviously, the subset B-lower approximation of X is the same set as the concept B-lower approximation of X. A concept B-upper approximation of the concept X is defined as follows:

$$\overline{B}X = \cup\{K_B(x)|x \in X, K_B(x) \cap X \neq \emptyset\} = \cup\{K_B(x)|x \in X\}.$$

The concept B-upper approximation of X is a subset of the subset B-upper approximation of X. Besides, the concept B-upper approximations are truly the

smallest B-definable sets containing X. For the decision table presented in Table 2, the concept A-lower and A-upper approximations are

$$\underline{A}\{1,2,4,8\} = \{1,2,4,8\},$$

$$\underline{A}\{3,5,6,7\} = \{3,7\},$$

$$\overline{A}\{1,2,4,8\} = \{1,2,4,6,8\},$$

$$\overline{A}\{3,5,6,7\} = \{3,4,5,6,7,8\}.$$

Note that for complete decision tables, all three definitions of lower approximations, singleton, subset and concept, coalesce to the same definition. Also, for complete decision tables, all three definitions of upper approximations coalesce to the same definition. This is not true for incomplete decision tables, as our example shows.

4 Rule Induction

In the first step of processing the input data file, the data mining system LERS (Learning from Examples based on Rough Sets) checks if the input data file is *consistent* (i.e., if the file does not contain conflicting examples). Table 1 is inconsistent because the fourth and the fifth examples are conflicting. For these examples, the values of all three attributes are the same (*high*, *yes*, *yes*), but the decision values are different, *yes* for the fourth example and *no* for the fifth example. If the input data file is inconsistent, LERS computes lower and upper approximations of all concepts. Rules induced from the lower approximation of the concept *certainly* describe the concept, so they are called *certain*. On the other hand, rules induced from the upper approximation of the concept describe the concept only *possibly* (or *plausibly*), so they are called *possible* [2].

The same idea of blocks of attribute-value pairs is used in a rule induction algorithm LEM2 (Learning from Examples Module, version 2), a component of LERS. LEM2 learns *discriminant description*, i.e., the smallest set of minimal rules, describing the concept. The option LEM2 of LERS is most frequently used since – in most cases – it gives best results. LEM2 explores the search space of attribute-value pairs. Its input data file is a lower or upper approximation of a concept, so its input data file is always consistent. In general, LEM2 computes a local covering and then converts it into a rule set. We will quote a few definitions to describe the LEM2 algorithm.

Let B be a nonempty lower or upper approximation of a concept represented by a decision-value pair (d, w). Set B *depends* on a set T of attribute-value pairs $t = (a, v)$ if and only if

$$\emptyset \neq [T] = \bigcap_{t \in T} [t] \subseteq B.$$

Set T is a *minimal complex* of B if and only if B depends on T and no proper subset T' of T exists such that B depends on T'. Let $\mathcal{T}$ be a nonempty collection

of nonempty sets of attribute-value pairs. Then $\mathcal{T}$ is a *local covering* of B if and only if the following conditions are satisfied:

(1) each member T of $\mathcal{T}$ is a minimal complex of B,

(2) $\bigcap_{t \in \mathcal{T}}[T] = B$, and

$\mathcal{T}$ is minimal, i.e., $\mathcal{T}$ has the smallest possible number of members.

The procedure LEM2 is presented below.

Procedure LEM2
(**input**: a set B,
output: a single local covering $\mathcal{T}$ of set B);
begin
$G := B$;
$\mathcal{T} := \emptyset$;
while $G \neq \emptyset$
begin
$T := \emptyset$;
$T(G) := \{t | [t] \cap G \neq \emptyset\}$;
while $T = \emptyset$ **or** $[T] \not\subseteq B$
begin
select a pair $t \in T(G)$ such that $|[t] \cap G|$
is maximum; if a tie occurs, select a pair $t \in T(G)$
with the smallest cardinality of $[t]$;
if another tie occurs, select first pair;
$T := T \cup \{t\}$;
$G := [t] \cap G$;
$T(G) := \{t | [t] \cap G \neq \emptyset\}$;
$T(G) := T(G) - T$;
end {while}
for each $t \in T$ **do**
if $[T - \{t\}] \subseteq$ B **then** $T := T - \{t\}$;
$\mathcal{T} := \mathcal{T} \cup \{T\}$;
$G := B - \cup_{T \in \mathcal{T}}[T]$;
end {while};
for each $T \in \mathcal{T}$ **do**
if $\bigcup_{S \in \mathcal{T} - \{T\}}[S] = B$ **then** $\mathcal{T} := \mathcal{T} - \{T\}$;
end {procedure}.

MLEM2 is a modified version of the algorithm LEM2. The original algorithm LEM2 needs discretization, a preprocessing, to deal with numerical attributes. The MLEM2 algorithm can induce rules from incomplete decision tables with numerical attributes. Its previous version induced certain rules from incomplete decision tables with missing attribute values interpreted as lost and with numerical attributes. Recently, MLEM2 was further extended to induce both certain and possible rules from a decision table with some missing attribute values being lost and some missing attribute values being "do not care" conditions, while

some attributes may be numerical. Rule induction from decision tables with numerical attributes will be described in the next section. In this section we will describe a new way in which MLEM2 handles incomplete decision tables.

Since all characteristic sets $K_B(x)$, where $x \in U$, are intersections of attribute-value pair blocks for attributes from B, and for subset and concept definitions of B–lower and B–upper approximations are unions of sets of the type $K_B(x)$, it is most natural to use an algorithm based on blocks of attribute-value pairs, such as LEM2 [1], [4] for rule induction.

First of all let us examine rule induction usefulness for the three different definition of lower and upper approximations: singleton, subset and concept. The first observation is that singleton lower and upper approximations should not be used for rule induction. Let us explain that on the basis of our example of the decision table from Table 2. The singleton A-lower approximation of the concept {1, 2, 4, 8} is the set {1, 2, 4}. Our expectation is that we should be able to describe the set {1, 2, 4} using given interpretation of missing attribute values, while in the rules we are allowed to use conditions being attribute-value pairs. However, this is impossible, because, as follows from the list of all sets $K_A(x)$ there is no way to describe case 1 not describing at the same time case 8, but $\{1, 8\} \not\subseteq \{1, 2, 4\}$. Similarly, there is no way to describe the singleton A-upper approximation of the concept {3, 5, 6, 7}, i.e., the set {3, 5, 6, 7, 8}, since there is no way to describe case 5 not describing, at the same time, cases 4 and 8, however, $\{4, 5, 8\} \not\subseteq \{3, 5, 6, 7, 8\}$. On the other hand, both subset and concept A-lower and A-upper approximations are unions of the characteristic sets of the type $K_A(x)$, therefore, it is always possible to induce certain rules from subset and concept A-lower approximations and possible rules from concept and subset A-upper approximations. Subset A-lower approximations are identical with concept A-lower approximations so it does not matter which approximations we are going to use. Since concept A-upper approximations are subsets of the corresponding subset A-upper approximations, it is more feasible to use concept A-upper approximations, since they are closer to the concept X, and rules will more precisely describe the concept X. Moreover, it better fits into the idea that the upper approximation should be the smallest set containing the concept. Therefore, we will use for rule induction only concept lower and upper approximations.

In order to induce certain rules for our example of the decision table presented in Table 2, we have to compute concept A-lower approximations for both concepts, {1, 2, 4, 8} and {3, 5, 6, 7}. The concept lower approximation of {1, 2, 4, 8} is the same set {1, 2, 4, 8}, so we are going to pass to the procedure LEM2 as the set B. Initially $G = B$. The set $T(G)$ is the following set {(Temperature, high), (Temperature, very_high), (Temperature, normal), (Headache, yes), (Nausea, no), Nausea, yes)}.

For three attribute-value pairs from $T(G)$, namely, (Temperature, high), (Headache, yes) and (Nausea, yes), the following value

$$[(attribute, value)] \cap G$$

is maximum. The second criterion, the smallest cardinality of [(attribute, value)], indicates (Temperature, high), (Headache, yes) (in both cases that cardinality is equal to four). The last criterion, "first pair", selects (Temperature, high). Thus $T = \{$(Temperature, high)$\}$, $G = \{1, 4, 8\}$, and the new $T(G)$ is equal to {(Temperature, very_high), (Temperature, normal), (Headache, yes), (Nausea, no), Nausea, yes)}.

Since $[(Temperature, high)] \not\subseteq B$, we have to perform the next iteration of the inner WHILE loop. This time (Headache, yes) will be selected, the new T = {(Temperature, high), (Headache, yes)} and new G is equal to $\{4, 8\}$. Since $[T] = [(Temperature, high)] \cap [(Headache, yes)] = \{4, 8\} \subseteq B$, the first minimal complex is computed.

It is not difficult to see that we cannot drop any of these two attribute-value pairs, so $\mathcal{T} = \{T\}$, and the new G is equal to $B - \{4, 8\} = \{1, 2\}$.

During the second iteration of the outer WHILE loop, the next minimal complex T is identified as {(Temperature, very_high)}, so $\mathcal{T}$ = {{(Temperature, high), (Headache, yes)}, {(Temperature, very_high)}} and $G = \{1\}$.

We need one additional iteration of the outer WHILE loop, the next minimal complex T is computed as {(Temperature, high), (Nausea, no)}, and $\mathcal{T}$ = {{(Temperature, high), (Headache, yes)}, {(Temperature, very_high)}, {(Temperature, high), (Nausea, no)}} becomes the first local covering, since we cannot drop any of minimal complexes from $\mathcal{T}$. The set of certain rules, corresponding to $\mathcal{T}$ and describing the concept {1, 2, 4, 8}, is

(Temperature, high) & (Headache, yes) -> (Flu, yes),
(Temperature, very_high) -> (Flu, yes),
(Temperature, high) & (Nausea, no) -> (Flu, yes).

Remaining rule sets, certain for the second concept equal to {3, 5, 6, 7}, and both sets of possible rules are compute in a similar manner. Eventually, rules in the LERS format (every rule is equipped with three numbers, the total number of attribute-value pairs on the left-hand side of the rule, the total number of examples correctly classified by the rule during training, and the total number of training cases matching the left-hand side of the rule) are:
certain rule set:

2, 2, 2
(Temperature, high) & (Headache, yes) -> (Flu, yes)
1, 2, 2
(Temperature, very_high) -> (Flu, yes)
2, 2, 2
(Temperature, high) & (Nausea, no) -> (Flu, yes)
1, 2, 2
(Headache, no) -> (Flu, no)

and possible rule set:

1, 3, 4
(Headache, yes) -> (Flu, yes)
2, 2, 2
(Temperature, high) & (Nausea, no) -> (Flu, yes)
2, 1, 3
(Nausea, yes) & (Temperature, high) -> (Flu, no)
1, 2, 2
(Headache, no) -> (Flu, no)
1, 2, 3
(Temperature, normal) -> (Flu, no)

5 Other Approaches to Missing Attribute Values

So far we have used two approaches to missing attribute values, in the first one a missing attribute value was interpreted as lost, in the second as a "do not care" condition. There are many other possible approaches to missing attribute values, for some discussion on this topic see [7]. Our belief is that for any possible interpretation of a missing attribute vale, blocks of attribute-value pairs may be re-defined, a new characteristic relation may be computed, corresponding lower and upper approximations computed as well, and eventually, corresponding certain and possible rules induced.

As an example we may consider another interpretation for "do not care" conditions. So far, in computing the block for an attribute-value pair (a, v) we added all cases with value "*" to such block $[(a, v)]$. Following [7], we may consider another interpretation of "do not care conditions": If for an attribute a there exists a case x such that the corresponding value is a "do not care" condition, i.e., $\rho(x, a) = *$, then the corresponding case x should be included in blocks $[(a, v)]$ for all values v of attribute a with the same decision value as for x (i.e., we will add x only to members of the same concept to which x belongs). With this new interpretation of "*"s, blocks of attribute-value pairs for Table 2 are:

[(Temperature, high)] = {1, 4, 5, 8},
[(Temperature, very_high)] = {2, 8},
[(Temperature, normal)] = {6, 7},
[(Headache, yes)] = {2, 4, 6, 8},
[(Headache, no)] = {3, 7},
[(Nausea, no)] = {1, 3, 6},
[(Nausea, yes)] = {2, 4, 5, 7, 8}.

The characteristic set $K_B(x)$ for Table 2, a new interpretation of "*"s, and $B = A$, are:

$K_A(1) = \{1, 4, 5, 8\} \cap \{1, 3, 6\} = \{1, 8\}$,
$K_A(2) = \{2, 8\} \cap \{2, 4, 6, 8\} \cap \{2, 4, 5, 7, 8\} = \{2, 8\}$,
$K_A(3) = \{3, 7\} \cap \{1, 3, 6\} = \{3\}$,
$K_A(4) = \{1, 4, 5, 8\} \cap \{2, 4, 6, 8\} \cap \{2, 4, 5, 7, 8\} = \{4, 8\}$,

$K_A(5) = \{1, 4, 5, 8\} \cap \{2, 4, 5, 7, 8\} = \{4, 5, 8\}$,
$K_A(6) = \{6, 7\} \cap \{2, 4, 6, 8\} \cap \{1, 3, 6\} = \{6\}$,
$K_A(7) = \{6, 7\} \cap \{3, 7\} \cap \{2, 4, 5, 7, 8\} = \{7\}$, and
$K_A(8) = \{2, 4, 6, 8\}$.

The characteristic relation $R(B)$ is {(1, 1), (1, 8), (2, 2), (2, 8), (3, 3), (4, 4), (4, 8), (5, 4), (5, 5), (5, 8), (6, 6), (7, 7), (8, 2), (8, 4), (8, 6), (8, 8)}. Then we may define lower and upper approximations and induce rules using a similar technique as in the previous section.

6 Incomplete Decision Tables with Numerical Attributes

An example of an incomplete decision table with a numerical attribute is presented in Table 3.

Table 3. An incomplete decision table with a numerical attribute

	Attributes			Decision
Case	Temperature	Headache	Nausea	Flu
1	98	?	no	yes
2	101	yes	yes	yes
3	?	no	no	no
4	99	yes	yes	yes
5	99	?	yes	no
6	96	yes	no	no
7	96	no	yes	no
8	*	yes	*	yes

Numerical attributes should be treated in a little bit different way as symbolic attributes. First, for computing characteristic sets, numerical attributes should be considered as symbolic. For example, for Table 3 the blocks of the numerical attribute Temperature are:

[(Temperature, 96)] = {6, 7, 8},
[(Temperature, 98)] = {1, 8},
[(Temperature, 99)] = {4, 5, 8},
[(Temperature, 101)] = {2, 8}.

Remaining blocks of attribute-value pairs, for attributes Headache and Nausea, are the same as for Table 2. The characteristic sets $K_B(x)$ for Table 3 and $B = A$ are:

$K_A(1) = \{1, 8\} \cap \{1, 3, 6, 8\} = \{1, 8\}$,
$K_A(2) = \{2, 8\} \cap \{2, 4, 6, 8\} \cap \{2, 4, 5, 7, 8\} = \{2, 8\}$,
$K_A(3) = \{3, 7\} \cap \{1, 3, 6, 8\} = \{3\}$,
$K_A(4) = \{4, 5, 8\} \cap \{2, 4, 6, 8\} \cap \{2, 4, 5, 7, 8\} = \{4, 8\}$,
$K_A(5) = \{4, 5, 8\} \cap \{2, 4, 5, 7, 8\} = \{4, 5, 8\}$,

$K_A(6) = \{6, 7, 8\} \cap \{2, 4, 6, 8\} \cap \{1, 3, 6, 8\} = \{6, 8\}$,
$K_A(7) = \{6, 7, 8\} \cap \{3, 7\} \cap \{2, 4, 5, 7, 8\} = \{7\}$, and
$K_A(8) = \{2, 4, 6, 8\}$.

The characteristic relation $R(B)$ is {(1, 1), (1, 8), (2, 2), (2, 8), (3, 3), (4, 4), (4, 8), (5, 4), (5, 5), (5, 8), (6, 6), (6, 8), (7, 7), (8, 2), (8, 4), (8, 6), (8, 8)}. For the decision presented in Table 3, the concept A-lower and A-upper approximations are

$$\underline{A}\{1, 2, 4, 8\} = \{1, 2, 4, 8\},$$
$$\underline{A}\{3, 5, 6, 7\} = \{3, 7\},$$
$$\overline{A}\{1, 2, 4, 8\} = \{1, 2, 4, 6, 8\},$$
$$\overline{A}\{3, 5, 6, 7\} = \{3, 4, 5, 6, 7, 8\}.$$

For inducing rules, blocks of attribute-value pairs are defined differently than in computing characteristic sets. MLEM2 has an ability to recognize integer and real numbers as values of attributes, and labels such attributes as numerical. For numerical attributes MLEM2 computes blocks in a different way than for symbolic attributes. First, it sorts all values of a numerical attribute, ignoring missing attribute values. Then it computes cutpoints as averages for any two consecutive values of the sorted list. For each cutpoint c MLEM2 creates two blocks, the first block contains all cases for which values of the numerical attribute are smaller than c, the second block contains remaining cases, i.e., all cases for which values of the numerical attribute are larger than c. The search space of MLEM2 is the set of all blocks computed this way, together with blocks defined by symbolic attributes. Starting from that point, rule induction in MLEM2 is conducted the same way as in LEM2. Note that if in a rule there are two attribute value pairs with overlapping intervals, a new condition is computed with the intersection of both intervals. Thus, the corresponding blocks for Temperature are:

[(Temperature, 96..97)] = {6, 7, 8},
[(Temperature, 97..101)] = {1, 2, 4, 5, 8},
[(Temperature, 96..98.5)] = {1, 6, 7, 8},
[(Temperature, 98.5..101)] = {2, 4, 5, 8},
[(Temperature, 96..100)] = {1, 4, 5, 6, 7, 8},
[(Temperature, 100..101)] = {2, 8}.

Remaining blocks of attribute-value pairs, for attributes Headache and Nausea, are the same as for Table 2. Using the MLEM2 algorithm, the following rules are induced from the concept approximations:

certain rule set:

2, 3, 3
(Temperature, 98.5..101) & (Headache, yes) -> (Flu, yes)
1, 2, 2
(Temperature, 97..98.5) -> (Flu, yes)
1, 2, 2
(Headache, no) -> (Flu, no)

possible rule set:

1, 3, 4
(Headache, yes) -> (Flu, yes)
2, 2, 3
(Temperature, 96..98.5) & (Nausea, no) -> (Flu, yes)
2, 2, 4
(Temperature, 96..100) & (Nausea, yes) -> (Flu, no)
1, 2, 3
(Temperature, 96..97) -> (Flu, no)
1, 2, 2
(Headache, no) -> (Flu, no)

7 Conclusions

It was shown in the paper that the idea of attribute-value pair blocks is an extremely useful tool. That idea may be used for computing characteristic relations for incomplete decision tables; in turn, characteristic sets are used for determining lower and upper approximations. Furthermore, the same idea of

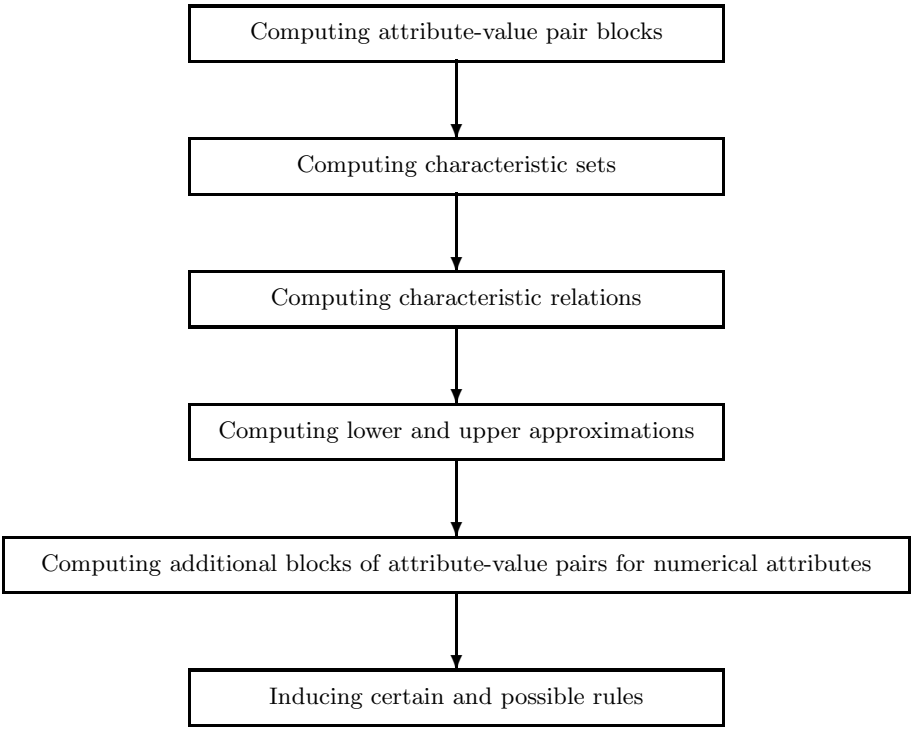

Fig. 1. Using attribute-value pair blocks for rule induction from incomplete decision tables

attribute-value pair blocks may be used for rule induction, for example, using the MLEM2 algorithm. The process is depicted in Figure 1.

Note that it is much more convenient to define the characteristic relations through the two-stage process of determining blocks of attribute-value pairs and then computing characteristic sets than to define characteristic relations, for every interpretation of missing attribute values, separately.

For completely specified decision tables any characteristic relation is reduced to an indiscernibility relation. Also, it is shown that the most useful way of defining lower and upper approximations for incomplete decision tables is a new idea of concept lower and upper approximations. Two new ways to define lower and upper approximations for incomplete decision tables, called subset and concept, and the third way, defined previously in a number of papers [9], [10], [13], [14], [15] and called here singleton lower and upper approximations, are all reduced to respective well-known definitions of lower and upper approximations for complete decision tables.

References

1. Chan, C.C. and Grzymala-Busse, J.W.: On the attribute redundancy and the learning programs ID3, PRISM, and LEM2. Department of Computer Science, University of Kansas, TR-91-14, December 1991, 20 pp.
2. Grzymala-Busse, J.W.: Knowledge acquisition under uncertainty – A rough set approach. *Journal of Intelligent & Robotic Systems* **1** (1988), 3–16.
3. Grzymala-Busse, J.W.: On the unknown attribute values in learning from examples. Proc. of the ISMIS-91, 6th International Symposium on Methodologies for Intelligent Systems, Charlotte, North Carolina, October 16–19, 1991. Lecture Notes in Artificial Intelligence, vol. 542, Springer-Verlag, Berlin, Heidelberg, New York (1991) 368–377.
4. Grzymala-Busse, J.W.: LERS – A system for learning from examples based on rough sets. In Intelligent Decision Support. Handbook of Applications and Advances of the Rough Sets Theory, ed. by R. Slowinski, Kluwer Academic Publishers, Dordrecht, Boston, London (1992) 3–18.
5. Grzymala-Busse., J.W.: MLEM2: A new algorithm for rule induction from imperfect data. Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU 2002, July 1–5, Annecy, France, 243–250.
6. Grzymala-Busse, J.W.: Rough set strategies to data with missing attribute values. Workshop Notes, Foundations and New Directions of Data Mining, the 3-rd International Conference on Data Mining, Melbourne, FL, USA, November 19–22, 2003, 56–63.
7. Grzymala-Busse, J.W. and Hu, M.: A comparison of several approaches to missing attribute values in data mining. Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing RSCTC'2000, Banff, Canada, October 16–19, 2000, 340–347.
8. Grzymala-Busse, J.W. and A. Y. Wang A.Y.: Modified algorithms LEM1 and LEM2 for rule induction from data with missing attribute values. Proc. of the Fifth International Workshop on Rough Sets and Soft Computing (RSSC'97) at the Third Joint Conference on Information Sciences (JCIS'97), Research Triangle Park, NC, March 2–5, 1997, 69–72.

9. Kryszkiewicz, M.: Rough set approach to incomplete information systems. Proceedings of the Second Annual Joint Conference on Information Sciences, Wrightsville Beach, NC, September 28–October 1, 1995, 194–197.
10. Kryszkiewicz, M.: Rules in incomplete information systems. *Information Sciences* **113** (1999) 271–292.
11. Pawlak, Z.: Rough Sets. *International Journal of Computer and Information Sciences* **11** (1982) 341–356.
12. Pawlak, Z.: Rough Sets. Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht, Boston, London (1991).
13. Stefanowski, J.: Algorithms of Decision Rule Induction in Data Mining. Poznan University of Technology Press, Poznan, Poland (2001).
14. Stefanowski, J. and Tsoukias, A.: On the extension of rough sets under incomplete information. Proceedings of the 7th International Workshop on New Directions in Rough Sets, Data Mining, and Granular-Soft Computing, RSFDGrC'1999, Ube, Yamaguchi, Japan, November 8–10, 1999, 73–81.
15. Stefanowski, J. and Tsoukias, A.: Incomplete information tables and rough classification. *Computational Intelligence* **17** (2001) 545–566.
16. Yao, Y.Y.: Two views of the theory of rough sets in finite universes. *International J. of Approximate Reasoning* **15** (1996) 291–317.
17. Yao, Y.Y.: Relational interpretations of neighborhood operators and rough set approximation operators. *Information Sciences* **111** (1998) 239–259.
18. Yao, Y.Y.: On the generalizing rough set theory. Proc. of the 9th Int. Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (RSFDGrC'2003), Chongqing, China, October 19–22, 2003, 44–51.