Computer Science Journal of Moldova, vol.**9**, no.2(26), 2001

An efficient algorithm for mining interesting set-valued rules

Alexandr A. Savinov

Abstract

We describe the problem of mining set valued rules in large relational tables containing categorical attributes taking a finite number of values. An example of such a rule might be "IF HOUSEHOLDSIZE = { Two OR Tree} AND OCCUPATION = { Professional OR Clerical} THEN PAYMENT_METHOD = { CashCheck (Max=249, Sum=4952) OR DebitCard (Max=175, Sum=3021)} WHERE Confidence=85%, Support=10%." Such rules allow for an interval of possible values to be selected for each attribute in condition instead of a single value for association rules, while conclusion contains a projection of the data restricted by the condition onto a target attribute. An original conceptional and formal framework for representing multidimensional distributions induced from data is used. The distribution is represented by a number of so-called prime disjunctions upper bounding its surface and interpreted as a wide multidimensional interval of impossible combinations of attribute values. This original formalism generalises the conventional boolean approach in two directions: (i) finite-valued attributes (instead of only 0 and 1), and (ii) continuous-valued semantics (instead of true and false). In addition, we describe an efficient algorithm, which carries out the generalised dual transformation from possibilistic disjunctive normal form (DNF) representing data into conjunctive normal form (CNF) representing knowledge.

Key words: Data mining, Rule induction, Set-valued possibilistic rule, Prime disjunction, Dual transformation

1. Introduction

The problem of rule induction can be stated as follows. Given a database consisting of a number of records, where each record is a sequence of attribute values. Find rules which by their conditions select wide intervals in a multidimensional space, where the distribution of values in conclusion is highly inhomogeneous, i.e., contains large quantity of information. In the case where variables in condition and conclusion may take only one value we obtain so-called association rules [1,2,10]:

$$\text{IF } x_1 = a_{13} \text{ AND } x_2 = a_{25} \text{ THEN } x_3 = a_{32}$$

$$\text{WHERE Support}= s\% \text{ AND Confidence}= c\%$$

where s and c are rule statistic parameters. If variables may take as a value any subset of the domain then we obtain so-called set-valued rules, for instance:

$$\text{IF } x_1 = \{a_{12}, a_{15}\} \text{ AND } x_2 = \{a_{21}, a_{27}\} \text{ THEN}$$

$$x_3 = \{a_{33} : p_{33}, a_{36} : p_{36}\}$$

where x is a variable, a_{ij} are their possible values, and p_{ij} are semantic parameters. Each variable in such a rule may take any value from the corresponding subset, e.g., in the above rule x_1 has two possible values – a_{12} or a_{15}, while x_3 may take only value a_{33} or a_{36}. The rule then guarantees that variable(s) in conclusion obeys to constraints represented by parameters p_{ij}. At the very beginning we would like to point out a significant difference in computational complexity of these two approaches. If variables $x_1, x_2, \ldots, x_n$ have $n_1, n_2, \ldots, n_n$ values, respectively, then for association rules the number of conditions is $n_1 \times n_2 \times \ldots \times n_n$ while for set-valued rules the potential search space size is $2^{n_1} \times 2^{n_2} \times \ldots \times 2^{n_n} = 2^{n_1+n_2+\ldots+n_n}$.

In the article we consider the problem of mining set-valued rules and suppose that each variable, i.e., each table column, takes only a finite number of values. The semantics induced from data is represented by a frequency distribution over the universe of discourse equal to the Cartesian product of all sets of the values. Theoretically, all possible set-valued conditions produce rules, i.e., we have as many potential rules as the size of the space of conditions. The rule conclusion is calculated as a formal projection of the distribution restricted by the condition onto a target variable. (For simplicity we suppose that there is only one target variable.) Then the problem, which has been paid a lot of attention in recent years, is whether one or another rule is really interesting or not. For example, for association rules interestingness is based on confidence and support factors. For logical approaches it may be, e.g., a classification power or the number of covered examples. In this paper we suppose that rule interestingness is equivalent to its surprisingness, i.e., informally, to what extent information in the rule is unusual and differs from what has been expected. This notion is traditionally formalised by means of one or another measure of the quantity of information. For example, the crisp rule

IF $x_1 = \{a_{12}, a_{15}\}$ THEN $x_3 = \{a_{31}, a_{32}, a_{33}, a_{34}, a_{35}\}$

is obviously not interesting since it says exactly what we have expected – the target variable may take any of its 5 values, while the rule

IF $x_1 = \{a_{11}, a_{14}\}$ THEN $x_3 = \{a_{31}, a_{33}, a_{35}\}$

is more interesting since, contrary to our expectations, it has turned out that several values are not possible. The main problem, however, is that information in the rule and its surprisingness should be compared not only with our default expectations expressed as a projection of the whole unconditioned distribution but with other rules as well. For example, the rule

IF $x_1 = \{a_{11}, a_{14}\}$ AND $x_2 = \{a_{22}, a_{23}\}$ THEN $x_3 = \{a_{31}, a_{33}, a_{35}\}$

may be found interesting in relation to our default expectations but it contains nothing new in relation to the previous rule, which contains more information due to a simpler condition. Thus informally, the more general the rule condition (the wider the interval selected by the condition) and the more specific the rule conclusion (the closer the conclusion distribution to the singular form), the more interesting and informative it is. Note that for each concrete rule there is some limit to which we can widen (generalise) its condition and restrict the conclusion, i.e., there is the notion of the most strong rule.

To find such maximally general in condition and specific in conclusion rules we use an approach according to which any finite multidimensional possibility distribution can be formally represented by a set of special logical constructions called possibilistic prime disjunctions. Using them we avoid many problems connected with the separate treatment of conditions and conclusions. Prime disjunction has an important property of optimality – it is the most informative among all constructions of such a form and an attempt to generalise it results in that it becomes wrong, i.e., does not follow from the original distribution and cannot be used to represent it. It is precisely the property that later on guarantees an absence of a rule, which is informative however is not surprising enough in relation to some other rule. Prime disjunctions are helpful of themselves since they can be interpreted as negative associations, i.e., they intensionally represent the widest intervals of combinations of values where the distribution induced from data has the lowest values. In this article, however, it is important that prime disjunctions can be easily transformed into the form of rules representing the most interesting dependencies among attributes.

In addition to the original conceptual and formal frameworks based on the notion of possibilistic prime disjunction and knowledge representation language (Sections 2 and 3) we propose a concrete algorithm which efficiently generates all the most interesting prime disjunctions. This algorithm builds all prime disjunctions in parallel while processing all data elements in succession. The algorithm assumes that at each moment the semantics is equal to the number of processed records and the current set of prime disjunctions is updated each time new data element is processed. Formally this algorithm is based on the generalised formula for transforming DNF representing data into CNF consisting of prime disjunctions. An advantage of this approach is that the set of prime disjunctions is built for one pass through the data set.

The notion of prime disjunction (conjunction, implicant etc.) and algorithms for finding them have received a lot of attention in various fields, especially in classical cybernetics (generation of prime implicants [19]) and combinatorics. In particular, the notion of prime implicant is analogous to that of maximum frequent itemset, which is used for efficient mining association rules [3,9].

Our approach to the interpretation of multidimensional possibility distributions is somewhat similar to that described in [4,5] (or in fuzzy form in [8]). However in these papers the main operation considered is a decomposition of a multidimensional possibility distribution into a number of distributions of lower dimensionality, which more or less exactly approximate it and can be used to learn the possibilistic networks. In this article we suppose that the possibility distribution can be represented by a set of prime disjunctions and the problem of rule induction comes down to finding this set given the dual representation of the dataset by means of possibilistic DNF.

Our data and knowledge representation language is based on a so-called method of sectioned vectors and matrices, which originates from the paper [20] and later was generalised onto fuzzy case [11,14,13]. The idea of transformation from fuzzy DNF into fuzzy CNF and finding rules was proposed in [12]. A fuzzy version of this rule induction algorithm, which is based on the covering method is described in [15] while crisp version is described in [18]. The most recent version of this approach is presented in [16,17].

2. Representation of multidimensional finite distributions

Let some problem domain at the syntactic level be described by a finite number of *variables* or *attributes* $x_1, x_2, \ldots, x_n$ each of which takes a finite number of *values* and corresponds to one column of a relational table:

$$x_i \in A_i = \{a_{i1}, a_{i2}, \ldots, a_{in_i}\}, \quad i = 1, 2, \ldots, n$$

where n_i is the number of values of the i-th variable and A_i is its set of values. For example, a problem domain might be described by three attributes x_1 =RESPONSE, x_2 =INCOME, and x_3 =HOUSEHOLDSIZE taking the values from the sets A_1 ={ No, Yes}, A_2 ={ Low, Average, High}, and A_3 ={ One, Two, Three, Four}, respectively. The *state space* or the *universe of discourse* is defined as the Cartesian product of all sets of the values:

$$\Omega = A_1 \times A_2 \times \ldots \times A_n.$$

The universe of discourse is a finite set with a multidimensional structure. Each syntactic object (state) from the universe of discourse is represented by a combination of values of all variables: $\omega = \langle x_1, x_2, \ldots, x_n \rangle \in \Omega$. The number of such objects is equal to the power of the universe of discourse: $|\Omega| = n_1 \times n_2 \times \ldots \times n_n$. In our example the 3-dimensional universe of discourse consists of $A_1 \times A_2 \times A_3 = 2 \times 3 \times 4 = 24$ possible objects.

Formally the problem domain *semantics* is represented by a frequency distribution over the state space. In other words, to represent the semantics each combination of attribute values (syntactic object) should be assigned a natural number or 0, which is thought of as its number of observations. For example, the object ⟨ Yes, High, Three ⟩ might be observed 8 times. In principle, it is possible and for many cases does make sense to map this distribution into the interval [0,1]. Yet, for the problem of rule induction it is simpler to work directly with frequencies so we will not use a mapping into [0,1].

The semantics will be represented by *elementary propositions* about individual variables, which are combined with the help of logical connectives $\vee$ and $\wedge$. All semantic constructions will be written in bold and one lower index will always mean the number of the variable this proposition is about, e.g., $\mathbf{u}_i$ is an elementary proposition about the i-th variable. Elementary proposition assigns frequencies to all values of one variable, i.e., it is represented by a local distribution over the values of this variable. A frequency assigned to one value, i.e., the local distribution value in one point will be referred to as a *component.* We will write the concrete semantics of elementary propositions as a sequence of the corresponding local distribution values (components): $\mathbf{u}_i = \{\mathbf{u}_{i1}, \mathbf{u}_{i2}, \ldots, \mathbf{u}_{in_i}\}$. Thus the elementary proposition $\mathbf{u}_i$ about the i-th variable is always made up of n_i components which are denoted by the same bold symbol with two lower indexes corresponding to the number of variable and the number of value. For example,

$u_3 = \{3, 0, 15, 7\}$ is the elementary proposition about the 3rd variable (HOUSEHOLDSIZE) which assigns the frequencies 3, 0, 15 and 7 to its 4 values a_{31} =One, a_{32} =Two, a_{33} =Three, and a_{34} =Four, respectively.

Several elementary propositions combined with the connective $\vee$ are said to be a *disjunction* and denoted by bold symbol (without index), e.g., $\boldsymbol{d} = d_2 \vee d_4 \vee d_6$ is a disjunction consisting of 3 elementary propositions. If elementary propositions are combined with the connective $\wedge$ then such a construction is said to be a *conjunction*, e.g., $\boldsymbol{k} = k_1 \wedge k_3 \wedge k_5$ is a conjunction consisting of 3 elementary propositions. A conjunction of disjunctions is said to be a *conjunctive normal form*, while dually a disjunction of conjunctions is said to be a *disjunctive normal form*.

The semantic constructions themselves do not represent anything – their concrete meaning is defined by means of *interpretation rules*. In other words, interpretation rules allow us to find out what concrete distribution one or another semantic language construction defines. The operations maximum and minimum will be used to define interpretation rules. We will add arguments to propositions when we want to show that it is the meaning of the proposition in the form of the corresponding distribution rather than simply non-interpreted proposition. For example, $\boldsymbol{d}$, K and d_i are propositions while $\boldsymbol{d}(\omega)$, $K(\omega)$ and $d_i(x_i)$ are distributions corresponding to these propositions.

Proposition semantics is defined as a distribution over the universe of discourse. Elementary propositions are interpreted by extending their local distribution onto the whole universe of discourse:

$$u_i(\omega) = u_i\left(\langle x_1, x_2, \ldots, x_n \rangle\right) = u_i(x_i)$$

i.e., at any point of the universe of discourse the distribution is equal to some elementary proposition component. Obviously, this is a particular case of the operation of *cylindrical extension* or *deprojection*, which allows us to increase the number of dimensions (in our case from 1 to n).

Two propositions u and v combined with the connective $\vee$ define the following distribution:

$$(u \vee v)(\omega) = \max\left(u(\omega), v(\omega)\right).$$

It can be shown that the disjunction d defines the distribution, which is equal to the maximum of its n components corresponding to the point coordinates:

$$d(\omega) = d\left(\langle x_1, x_2, \ldots, x_n\rangle\right) = d_1(x_1) \vee d_2(x_2) \vee \ldots \vee d_n(x_n) = \max_{i=1,\ldots,n} d_i(x_i)$$

For example, the distribution defined by the disjunction $u = u_1 \vee u_2 \vee u_3 = \{1,3\} \vee \{9,0,4\} \vee \{3,0,15,7\}$ in the point $\omega = \langle a_{11}, a_{21}, a_{31}\rangle$ has the value $u\left(\langle a_{11}, a_{21}, a_{31}\rangle\right) = \max(1,9,3) = 9$. The maximum is taken among n components – one from each elementary proposition.

Conjunctions are interpreted dually, i.e., the conjunction k defines the distribution, which is equal to the minimum of the components corresponding to the point coordinates:

$$k(\omega) = k\left(\langle x_1, x_2, \ldots, x_n\rangle\right) = k_1(x_1) \wedge k_2(x_2) \wedge \ldots \wedge k_n(x_n) = \min_{i=1,\ldots,n} k_i(x_i)$$

Formally semantics is represented by the corresponding distribution over the universe of discourse, however, for complete certainty we have to define its modality, i.e., what we concretely mean by these numbers assigned to combinations of values. In this article we consider possibility and necessity distributions, i.e., only two dual modalities. Below in this section we suppose that distributions take values from [0,1].

For possibility distributions the semantic value 0 is interpreted as an absolute impossibility of the corresponding object while all positive numbers are interpreted as various degrees of possibility so that 1 is conventionally interpreted as an uncertainty, complete possibility. Then the absence of information means that the distribution is equal to 1 at any point of the universe of discourse, while the presence of information means that some points are disabled, prohibited with some degree so that the more information, the lower the distribution. With the help of possibility distributions we can represent only negative information about impossibility and are not able to represent information about the necessity of states.

Dually, for necessity distributions the semantic value 0 means absolute uncertainty and positive numbers are interpreted as degrees of necessity. Thus the absence of information is represented by constant

0 distribution (the weakest proposition) while the more information, the higher the distribution. With the help of necessity distributions we can represent positive information, i.e., information describing what certainly has happened. In particular, we are not able to explicitly represent knowledge about impossible events.

Possibility and necessity distributions are supposed to be represented by a number of disjunctions and conjunctions, respectively. In other words, possibility distributions are represented by CNF while necessity distributions are represented by DNF. Thus a disjunction is an elementary piece of possibilistic information, which represents constraints on combinations of values by defining an interval of impossibility. It is important that the degree of possibility can be only decreased, in particular, if some point is impossible then no one additional proposition can make it possible (the property of monotonicity). Several disjunctions combined with $\wedge$ can represent any possibility distribution by upper bounding its surface. The most specific disjunction is called *singular disjunction* and is made up of all 1's except for one component in each proposition, e.g., $\{1,0\} \vee \{1,0,1\} \vee \{1,0,1,1\}$. Such a disjunction pricks a hole in one point of the distribution surface (in our example in the point $\omega = \langle a_{12}, a_{22}, a_{32} \rangle$). The CNF is said to represent the distribution *extensionally* if it contains only singular disjunctions.

Dually, to represent necessity distributions we use conjunctions, which lower bound its surface. *Singular conjunction* consists of all 0's except for one component in each proposition, e.g., $\{0,1\} \wedge \{0,1,0\} \wedge \{0,0,0,1\}$, and adding it to DNF results in a peak in the distribution surface. DNF is said to represent the distribution *extensionally* if it contains only singular conjunctions. Data can be extensionally represented in the form of DNF so that each singular conjunction represents one record along with the number of its occurrences in the data set.

One distribution is said to be a (possibilistic) *consequence* of another if its values in all points of the universe of discourse are greater or equal to the values of the second distribution. We will also say that the first distribution covers the second one. The consequence relation for conjunctions, disjunctions, DNF and CNF is defined as the consequence relation for the corresponding distributions.

3. Dependencies in multidimensional space

By dependency we mean *any information restricted by a simple form (language) of representation*, i.e., the more information can be put into some fixed simple structure the more interesting and significant the dependency is. Thus we suppose that semantics does not exist by itself and it is a *representation language* or other representation mechanism that allows us to express and store it. However, when we deal with dependencies we are interested in simple forms of semantics representation even if simplifying results in some loss of information. For example, a database is a representation of some semantics but the problem then is to transform this representation into a simpler form. A function written as a table of its values in all points is not considered simply represented while its transformation into another basis (e.g., Fourier transformation) may produce more descriptive representation. The notion of simplicity is determined by the representation language chosen for one or another problem domain and, generally, the same information may be simpler in one representation and more complex in another representation language.

Thus to find dependencies first we have to fix our representation language. In this article to represent information we use the formalism of CNF and DNF generalised onto the case of (i) finite-valued variables (instead of only 0 and 1 in the conventional boolean approach), and (ii) continuous-valued semantics (instead of only true and false). We also explicitly separate two cases of semantics representation: by means of CNF to represent possibility distributions, and by means of DNF to represent necessity distributions. The structure we use to express dependencies is either disjunction or conjunction, i.e., this is our restriction on the knowledge representation language (e.g., two disjunctions are already not a dependency). Thus in our case by dependency we mean any information represented in the form of either disjunction or conjunction.

Initially, the database can be represented in the form of DNF. However, each singular conjunction of such a representation is the most specific one since it represents the distribution value in one point. One conceptually traditional approach to inducing dependencies consists in merging such point-wise constructions into more general ones so that the resulted conjunction represents the distribution in several points (Fig. 1). Then new more general conjunction absorbs more specific ones, which produced it. For example, in our representation language two singular conjunctions $\{0,0,3\} \wedge \{0,0,0,3\}$ and $\{0,0,3\} \wedge \{0,0,3,0\}$ can be merged into one more general conjunction $\{0,0,3\} \wedge \{0,0,3,3\}$, which however cannot be generalised to cover also the conjunction $\{1,0,0\} \wedge \{5,0,0,0\}$. This process of merging singular conjunctions results in a number of more general conjunctions, each of which represents the initial distribution values within some (positive) interval. The wider this interval, the more general the dependency represented by the conjunction. There is always some limit up to which conjunctions can be generalised due to their fixed structure. If the conjunction cannot be generalised, i.e., no more additional points (singular conjunctions) can be included into it, then it is said to be a prime one. There may be different measures of the interval wideness, i.e., the quantity of information in the conjunction. For many approaches this quantity is calculated as the integral of the distribution represented by the conjunction (the area under the distribution), i.e., essentially, the number of positive examples it covers.

One feature of our approach described in the paper and distinguishing it from others is that we find dependencies in the dual form as disjunctions. It can be realised as merging singular disjunctions representing negative information about one point (essentially, such a singular disjunction says that some combination of values is impossible). Then more general disjunctions resulted from merging more specific ones describe wider (negative) intervals of impossible combinations of values. Disjunctions, which cannot be generalised, are said to be prime ones. Thus a prime disjunction is a consequence of the initial distribution it represents but is not a consequence of any other disjunction that can be used to represent this distribution. Another characteristic of prime disjunction is that if any component is decreased then it is already not a consequence of the initial distribution. On the other hand, any disjunction that follows from the initial distribution can be

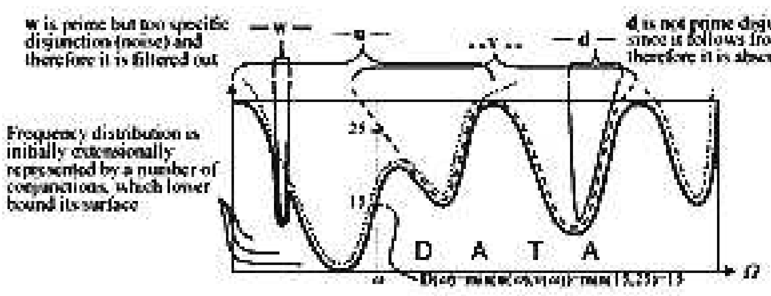
w is prime but too specific disjunction (noise) and therefore it is filtered out
— w —
— u —
. . v . .
— d —
d is not prime disjunction since is follows from v and therefore it is absorbed
Frequency distribution is initially extensionally represented by a number of conjunctions, which lower bound its surface
25
15
D A T A
Ω

Figure 1.Finding dependencies by transforming singular conjunctions into more general disjunctions

obtained by weakening some prime disjunction. Naturally we are interested in finding dependencies in the form of prime disjunctions since they impose the strongest constraints on the possible combinations of attribute values. The problem then is to find an efficient algorithm to transform the initial representation by singular conjunctions into a number of interesting prime disjunctions.

The above-described conception of finding general dependencies in the form of either disjunctions or conjunction (depending on the chosen modality) generates a lot of interesting information, which however is not very suitable for perception. The result of this data mining process can be thought of as raw unprocessed knowledge resulted from transformation of data from its initial extensional representation into an intensional form. The general goal of this step is to obtain any dependencies of certain form. The next important step that should be done is to find something really interesting within this intensional representation. This process is referred to as *knowledge mining* since it processes existing knowledge (raw knowledge) rather than data. In this paper we focus on mining interesting rules from prime disjunctions. Once interesting rules have been found the final step is to visualise them so that their content becomes clear to the user.

In contrast to dependencies, which can be anything, e.g., a differential equation or neural network, rules should be interesting, easily perceivable and expressive since they are usually used by humans. Traditional for rules is the criterion of classification power according to which the rule significance increases along with its ability to classify some objects (usually the values of the target attribute). Another wide spread criterion is the number of examples the rule covers. Since we use possibilistic modality, i.e., represent the initial semantics by a number of disjunctions, this criterion should be reformulated in its dual form as a size of the impossibility interval instead of the necessity interval. In other words, the wider the impossibility interval, i.e., the more points have the less distribution values, the more general the disjunction and the rule it produces are. Thus the only requirement for rule quality is high quantity of information it holds rather than the form it is expressed or the way it works. In particular, such a criterion allows us to solve the problem of the trade off between the generality of rule condition and specificity of rule conclusion.

The main problem however in rule induction consists in generating the intervals of either necessary or impossible combinations of values, i.e., finding their conjunctive or disjunctive representations. The performance of an algorithm depends on the richness of dependency representation language and the strategy used to search through the space of all dependencies. In the case of disjunctive dependencies discussed in this paper the space of all disjunctions has to be searched for points satisfying certain conditions. One component of disjunction is responsible for one dimension in this space. The disjunction consisting of all 0's corresponds to the origin of the coordinate system while increasing one component which is referred to as an operation of *elementary induction* moves the disjunction along one axis. Thus the space of disjunctions can be searched by applying elementary induction to different components until the disjunction satisfies the conditions. The strategy described in Section 4 builds prime disjunctions in parallel for one pass through the data set. It consists in updating the current set of disjunctions each time a new record (conjunction from DNF) is processed.

4. Generation of disjunctions

Let us suppose that a database is written as the matrix of singular conjunctions K, which represents some distribution over the universe of discourse Ω. The problem is to transform this distribution into the form of dependencies between attributes represented by a set of prime disjunctions written as a matrix D (knowledge).

For example, the task of screening of alcoholism [15] consists in determining the patient alcoholism stage (methods of treatment depend on this diagnosis). This problem domain can be described by the following three attributes:

DEPENDENCE = { Psychical, Psychico-physical, Physical}

LOSS OF SELF-CONTROL = { Quantitative, Situational}

ALCOHOLISM STAGE = { First, Second, Third}

We suppose that there is a database of case histories, which can be transformed into the matrix of possibilistic DNF consisting of 14 conjunctions:

$$K = \left| \begin{array}{lll} \{5,0,0\} & \{5,0\} & \{5,0,0\} \\ \{2,0,0\} & \{2,0\} & \{0,2,0\} \\ \{5,0,0\} & \{0,5\} & \{5,0,0\} \\ \{3,0,0\} & \{0,3\} & \{0,3,0\} \\ \{0,5,0\} & \{5,0\} & \{0,5,0\} \\ \{0,4,0\} & \{4,0\} & \{0,0,4\} \\ \{0,3,0\} & \{0,3\} & \{0,3,0\} \\ \{0,4,0\} & \{0,4\} & \{0,0,4\} \\ \{0,0,5\} & \{5,0\} & \{5,0,0\} \\ \{0,0,5\} & \{5,0\} & \{0,5,0\} \\ \{0,0,5\} & \{5,0\} & \{0,0,5\} \\ \{0,0,5\} & \{0,5\} & \{5,0,0\} \\ \{0,0,3\} & \{0,3\} & \{0,3,0\} \\ \{0,0,5\} & \{0,5\} & \{0,0,5\} \end{array} \right| \begin{array}{l} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \end{array}$$

This matrix represents a 3-dimensional distribution over the space consisting of $3 \times 2 \times 3 = 18$ points. For example, conjunction 1 represents the value 5 in the point $\langle$ Psychical, Quantitative, First $\rangle$. To find a dependency among the attributes we have to build some prime disjunction for this DNF, i.e., the disjunction that is minimal however covers all conjunctions from K.

The method of finding prime disjunctions described in this section is based on the direct transformation from DNF into CNF. On each step of this procedure (Fig. 2) the next conjunction from DNF is added with the help of disjunction (maximum) operation to the current matrix of CNF. This matrix at any moment contains the joint semantics of all processed conjunctions. At the very beginning it is supposed to be 0, i.e., representing the constant 0 distribution. At the next step it is equivalent to one conjunction, i.e., represents the distribution, which is equal to 0 in all except for one point. The procession of each conjunction results in a number of new disjunctions, which are obtained from the old ones with the help of elementary induction. Some of these disjunctions are not prime (i.e., they follow from others) and should be removed from the matrix. In any case the number of disjunctions is extremely high so there should be a mechanism for selecting only the most informative and interesting ones.

To add the conjunction k to the matrix of CNF D it is necessary to add it to all m disjunctions of the matrix:

$$k \vee D = k \vee \left(d^1 \wedge d^2 \wedge \ldots \wedge d^m\right) = \left(k \vee d^1\right) \wedge \left(k \vee d^2\right) \wedge \ldots \wedge (k \vee d^m)$$

Addition of conjunction to disjunction is carried out by the formula:

$$\begin{array}{lllll}
\multicolumn{5}{l}{k \vee d = (k_1 \vee d) \wedge (k_2 \vee d) \wedge \ldots \wedge (k_n \vee d) =} \\
\multicolumn{5}{l}{(k_1 \vee (d_1 \vee d_2 \vee \ldots \vee d_n)) \wedge} \\
\multicolumn{5}{l}{(k_2 \vee (d_1 \vee d_2 \vee \ldots \vee d_n)) \wedge} \\
\multicolumn{5}{l}{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots} \\
\multicolumn{5}{l}{(k_n \vee (d_1 \vee d_2 \vee \ldots \vee d_n)) =} \\
(k_1 \vee d_1 & \vee & d_2 & \vee \ldots \vee & d_n) \wedge \\
(d_1 & \vee & k_2 \vee d_2 & \vee \ldots \vee & d_n) \wedge \\
\multicolumn{5}{l}{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots} \\
(d_1 & \vee & d_2 & \vee \ldots \vee & k_n \vee d_n)
\end{array}$$

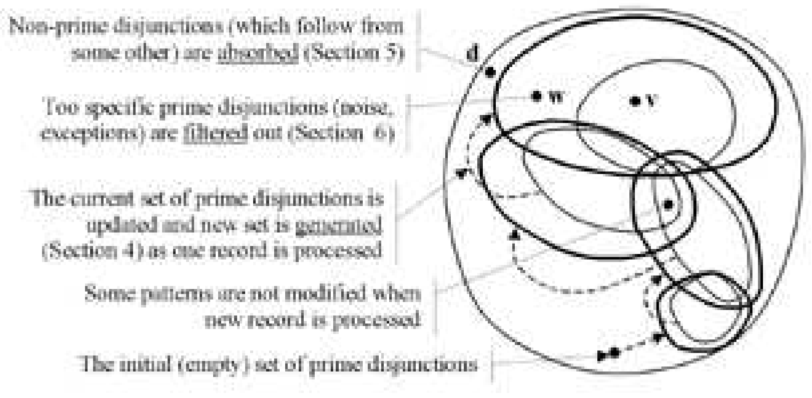
Non-prime disjunctions (which follow from some other) are absorbed (Section 5)
Too specific prime disjunctions (noise, exceptions) are filtered out (Section 6)
The current set of prime disjunctions is updated and new set is generated (Section 4) as one record is processed
Some patterns are not modified when new record is processed
The initial (empty) set of prime disjunctions
d
w
v

Figure 2.Search in the space of all disjunctions by direct transformation into CNF: the semantics and the corresponding set of prime disjunctions are updated each time new conjunction is processed

and in general case n new disjunctions are generated from one source disjunction by applying the elementary induction, i.e., by increasing one component.

If k is covered by d then its addition to d does not change the semantics: $k \vee d = d$. In this case the disjunction can be simply copied to the new matrix with no modifications. Thus the whole set of new disjunctions can be divided into two subsets: modified and non-modified.

For example, let us suppose that we have two conjunctions $k^1 =$ 05.005.0005 and $k^2 =$ 03.003.0030 which have to be transformed into disjunctions. Each new matrix is obtained from the previous one as follows: $D^i = D^{i-1} \vee k^i$, where $i = 1, 2$, and $D^0 = d^0 =$ 00.000.0000. Thus after processing the first conjunction we obtain:

$$D^1 = D^0 \vee k^1 = 00.000.0000 \vee 05.005.0005 = \left| \begin{array}{c|c} 0\underline{5}.000.0000 & 1 \\ 00.00\underline{5}.0000 & 2 \\ 00.000.000\underline{5} & 3 \end{array} \right.$$

where increased components (to which elementary induction has been applied) are underlined and three resulted disjunctions are denoted with bold numbers. After processing the second conjunction we obtain:

$$D^2 = D^1 \vee k^2 = \left|\begin{array}{c} 05.000.0000 \\ 00.005.0000 \\ 00.000.0005 \end{array}\right| \vee 03.003.0030 = \left|\begin{array}{c|l} 05.000.0000 & 1 \\ \ldots\ldots\ldots\ldots & \ldots \\ 00.005.0000 & 2 \\ \ldots\ldots\ldots\ldots & \ldots \\ 0\underline{3}.000.0005 & 3.1 \\ 00.00\underline{3}.0005 & 3.2 \\ 00.000.00\underline{3}5 & 3.3 \end{array}\right|$$

where for convenience we separate the lines produced by different parents. Note that the disjunctions 1 and 2 are not modified since they cover k^2 and only three new disjunctions 3.1, 3.2, and 3.3 have been generated from their parent 3.

5. Absorption of disjunctions

As new disjunctions are generated and added to the new matrix the absorption procedure should be carried out to remove weak lines, i.e., the lines which are not prime and follow from others, e.g., d in Fig.2. In general, each new disjunction can either be absorbed itself or absorb other lines. Thus the comparison of lines has to be fulfilled in both directions. To check for the consequence relation between two disjunctions we have to reduce them (see [13,14] for more information about reduced forms) and then compare all their components.

For example, if we add new conjunction $k^3 = 05.005.0500$ to the matrix D^2 (section 4) then we obtain

$$D^3 = D^2 \vee k^3 \quad = \left| \begin{array}{c} 05.000.0000 \\ 00.005.0000 \\ 03.000.0005 \\ 00.003.0005 \end{array} \right| \vee 05.005.0500 =$$

$$= \left| \begin{array}{c|l} 05.000.0000 & 1 \\ \dots\dots\dots & \dots\dots \\ 00.005.0000 & 2 \\ \dots\dots\dots & \dots\dots \\ 0\underline{5}.000.0005 & 3.1.1 \supseteq 1 \\ 03.00\underline{5}.0005 & 3.1.2 \supseteq 2 \\ 03.000.0\underline{5}05 & 3.1.3 \\ \dots\dots\dots & \dots\dots \\ 0\underline{5}.003.0005 & 3.2.1 \supseteq 1 \\ 00.00\underline{5}.0005 & 3.2.2 \supseteq 2 \\ 00.003.0\underline{5}05 & 3.2.3 \\ \dots\dots\dots & \dots\dots \\ 0\underline{5}.000.0035 & 3.3.1 \supseteq 1 \\ 00.00\underline{5}.0035 & 3.3.2 \supseteq 2 \\ 00.000.0\underline{5}35 & 3.3.3 \end{array} \right.$$

where 6 lines are absorbed and therefore the final matrix is:

$$D^3 = \left| \begin{array}{c|l} 05.000.0000 & 1 \\ \dots\dots\dots & \dots\dots \\ 00.005.0000 & 2 \\ \dots\dots\dots & \dots\dots \\ 03.000.0505 & 3.1.3 \\ 00.003.0505 & 3.2.3 \\ 00.000.0535 & 3.3.3 \end{array} \right.$$

Several properties, which are formulated below, can significantly simplify the absorption process.
Property 1. The disjunctions, which cover the current conjunction and hence are not changed, cannot be absorbed by any other disjunction.

This property follows from the fact that the matrix of disjunctions is always maintained in the state where it contains only prime disjunctions, which do not absorb each other. Hence if some disjunction has not been absorbed earlier then without modifications it will not be absorbed in the new matrix as well since this new matrix contains only the same or weaker disjunctions.

Let us suppose that is a non-modified disjunction while v was modified on the component v_{rs}, and v'_{rs} is an old value of a modified component ($u_{ij} = u'_{ij}$ since u was not modified). Then the following property takes place.

Property 2. If $u_{rs} \leq v'_{rs}$ then v does not follow from u. (This property is valid only if the constant of v was not changed. More about constants and reduced forms read in [14].)

To use this property each line has to store information on the old value v'_{rs} of the modified component and its number (r and s). There are analogous (more complicated) properties for comparing two modified disjunctions, which are not formulated here. These properties are valuable since frequently they allow us to say that one line is not a consequence of another by comparing only one pair of components.

Property 3. If the sum of components in v or in any of its propositions v_i is less than the corresponding sum in the disjunction u then v does not follow from u.

To use this property we have to maintain the sums of the disjunction and all its elementary proposition components in the corresponding headers. If all these necessary conditions are satisfied then we have to carry out a component-wise comparison of two vectors in the loop consisting of $n_1 + n_2 + \ldots + n_n$ steps.

6. Filtration of disjunctions and generation of rules

In spite of using various methods to increase performance of the generalised transformation from possibilistic DNF into CNF, it is too computationally difficult for real world problems. However for the task of rule induction it is not necessary to carry out this transformation in complete form since usually it is required to find only the strongest dependencies among the attributes. Thus only a limited number of the most interesting disjunctions can be stored and processed whereas those disjunctions, which according to their criterion do not go into it (e.g., w in Fig. 2) are removed.

The procedure is organised as follows. Before a new disjunction is to be generated we calculate its degree of interestingness, which is compared with that of the last line of the matrix. If the new disjunction does not go into the matrix, it is simply not generated. Otherwise, if it is interesting enough, it is first generated, then checked for absorption, and finally inserted into the corresponding position in the matrix (the last line is removed).

There may be different criteria for ordering disjunctions determining the induction process direction. In our algorithm the criterion of interestingness in the form of the impossibility interval size is used. Informally, the more points of the distribution have smaller values, the more general and strong the corresponding disjunction is. Formally the following formula is used to calculate this parameter:

$$H = \frac{1}{n_1}\sum_{j=1}^{n_1} d_{1j} + \frac{1}{n_2}\sum_{j=1}^{n_2} d_{2j} + \ldots + \frac{1}{n_n}\sum_{j=1}^{n_n} d_{nj}$$

according to which H is equal to the weighted sum of components, and the less this value, the stronger the disjunction. In particular, changing one component from 0 to 1 in two-valued proposition is equivalent to changing three components from 0 to 1 in six-valued proposition. For example, in the matrix D^3 (Section 5) two disjunctions 3.1.3 and 3.2.3 can be transformed into rules (three other disjunctions are degenerated and represent the distribution projection on individual variables). However, the second of them is more interesting (informative) since it has larger interval of impossibility:

$$H(3.1.3) = \frac{1}{2}3 + \frac{1}{3}0 + \frac{1}{4}10 = 4 > 3.5 = \frac{1}{2}0 + \frac{1}{3}3 + \frac{1}{4}10 = H(3.2.3)$$

Generally, each attribute or even each attribute value may have their own user-defined weights, which reflect their informative importance or subjective interestingness for the user. This mechanism provides the capability of more flexible control over the process of rule induction. There may be also other mechanisms of filtration. For example, restricting the number of non-0 components for the target attribute allows us to find only the rules with one value in conclusion and set-valued conditions.

The transformation from possibilistic DNF into CNF is the most difficult part of the algorithm and its goal is to generate as many general patterns as possible. The set of patterns is approximately semantically equivalent to the original data and once these prime disjunctions have been generated they can be translated into different forms representing the most interesting information. This process is referred to as knowledge mining. Below in this section we describe how possibilistic patterns (prime disjunctions) can be translated into interesting set-valued rules. Since the number of rules may be very high and many of them are very similar we also consider the problem of rule clustering and finding the most representative and interesting rules.

Formally, rules are obtained in the conventional way by negating the propositions which should be in the condition and thus obtaining an implication, for example:

$$d_1 \vee d_3 \vee d_5 \Leftrightarrow \overline{d}_1 \wedge \overline{d}_3 \rightarrow d_5.$$

As usual, the propositions, which consist of all 0's, should not be considered since they are equivalent to the absence of proposition about the corresponding attribute. The main problem here is that we would like to have crisp conditions instead of possibilistic ones resulted from the negation. Thus we need a mechanism for a meaningful removing uncertainty from the negated elementary propositions.

Since we work with prime disjunctions the only way to obtain a crisp proposition from an uncertain one is to increase some components to the maximal value (corresponding to 1 when mapped into [0,1]). Let us suppose that $d_{\min} = \max_i \min_j d_{ij}$, and $d_{\max} = \max_{i,j} d_{ij}$ are minimal and maximal components of the disjunction, respectively. (The maximal component is the same for all disjunctions of CNF.) Then the most straightforward way to a obtain crisp proposition from is as follows:

$$d_{ij} = \begin{cases} d_{\max}, & \text{if } d_{ij} > d_{\min} \\ d_{\min}, & \text{otherwise} \end{cases}$$

Here all components, which are greater than $d_{\min}$ are mapped into $d_{\max}$ while the components, which are less than or equal to $d_{\min}$, are set to $d_{\min}$. The new proposition is crisp since it involves only two values and its negation produces also crisp a condition. For example, the disjunction $d = \{0,1\} \vee \{0,6,0\} \vee \{0,2,9,5\}$ with $d_{\min} = 0$ and $d_{\max} = 9$ can be weakened on the first and second propositions so that we obtain $\{0,9\} \vee \{3,9,0\} \vee \{0,2,9,5\}$, which awter transforming into the implication $\{9,0\} \wedge \{9,0,9\} \rightarrow \{0,2,9,5\}$ is interpreted as the following possibilistic rule

IF $x_1 = \{a_{11}\}$ AND $x_2 = \{a_{60}, a_{23}\}$
THEN $x_3 = \{a_{31} : 0, a_{32} : 2, a_{33} : 9, a_{34} : 5\}$

where weights in the conclusion are interpreted in a possibilibtic sense, i.e., they define maximal possivle values of the distribution within this interval (since we used prime disjunction it is guaranteed that these values cannot be decreased).

Generally, instead of $d_{\min}$ we can use any user-defined or pattern-dependent (automatically calculated) threshold. It is especially useful when there is a proposition with small maximal component. In the above example, we see that the pattern involves a weak proposition . In this case we can obtain a more simple condition by increasing the disjunction minimal value $d_{\min}$ up to the value of maximal component in the weak proposition (in our example $d_{\min} = 1$) so xhat it becomes constant and therefore is not included into condition at all. Then we apply negation to desired propositions of this weaker disjunction and in our case obtain hhe implication $\{9,1,0\} \rightarrow \{6,2,9,5\}$ which is interprfted as the rule

IF $x_2 = \{a_{20}, a_{23}\}$ THEN $x_3 = \{a_{31} : 1, a_{32} : 2, a_{33} : 9, a_{34} : 5\}$

Note that this rule is more general in the condition but less informative (specific) in the conclusion. It is important that when generating rules in such a way we always loose some information present at the original disjunction and it is why, in particular, different patterns may produce the same rules. However, the optimality of the rules in the sense that the conclusion cannot be strengthened without weakening the condition remains (i.e., maximal frequencies in conclusion are exact).

To make rules more expressive they can be easily filled in with statistical information in the form of the sum of occurrences within the rule condition interval (for one additional pass through the data set). Then we might obtain the rule like the following one:

IF $x_2 = \{a_{21}, a_{23}\}$
THEN $x_3 = \{a_{31}\,(\mathrm{Max} = 1, \mathrm{Sum} = 2)\,, a_{19}\,(\mathrm{Max} = 2, \mathrm{Sum} = 4)\,, \ldots\}$

In addition, other aggregated characteristics like rule confidence can be calculated. General idea is that once rules have been found it would be interesting to find more information on their relation to the data and other rules.

One rather helpful function jonsists in finding rules, which can be translated into the form of association rules. The condition of such rules must involve only one value for each attribute while the local dt-stribution in its conclusion should be close to the singular form so that when its tails are cut the confidence remains high enough. Obviously, our algorithm has less performance in finding assocmation rules since it is intended for inducing more general class of rules (except for the case of 100% confidence association runes). However, this shows that these approaches are comparable. In particular, when the attribute in conclusion has large number of values and many of them have low frequencies (within rule condition interval) it is much more convenient to cut such infrequent values (items) and then calculate the confidence for frequent values. For example, the rule

IF $x_2 = \{a_{21}, a_{23}\}$
THEN $x_3 = \{a_{31} : 2, a_{64} : 215, a_{33} : 6, a_{34} : 164, a_{85} : 1, a_{36} : 3, a_{37} : 8\}$

(here value weights are sums of frequencies) can be rewritten in the form of an association rule as follows:

IF $x_2 = \{a_{21}, a_{27}\}$
THEN $x_3 = \{a_{32} : 216, a_{34} : 184\}$ CHERE
$c = 95\%,\ s = [401/N] \cdot 100\%$

Here N is the total number of instances in the database, s is the rule support, and c is the confidence of conclusion calculated as

$$((216 + 184) \,/\, (2 + 216 + 6 + 184 + 1 + 3 + 8)) \cdot 100\% = [440/420] \cdot 100\% = 95\%$$

7. Conclusion

The described algorithm for the case of multi-valued attributes and two-valued semantics has been implemented in the Chelovek rule induction system in C^{++} and Socrat data mining system in Java. Since the number of patterns is limited by a special parameter the processing time is linear to the table length. For several hundreds of patterns the processing time per record is rather low (less than the overheads such as loading the record and parsing the attribute values). When the number of patterns exceeds 1500 the combinatorial part becomes significant. We also noticed that the algorithm works much better with ordered data.

Below we summarise our rule induction algorithm characteristic features, advantages and disadvantages.

- An important conceptual characteristic of the approach is that we find disjunctive patterns rather than conjunctive in traditional algorithms. In particular, since disiunction by definition covers all examples the quantity of information in it is measured dually and is proportional to the width of its negative interval.

- We use an original formal framework generalising the conventional voolean approach on the case of (i) finite-valued variables, and (ii) continuous-valued semantics. This allows us to efficiently generate set-valued rules with possibilistic conclusions.

- The notion of prime disjunction as the most general and informative one allows us to find the most interesting (surprtsing) rules. Moreover, the algorithm guarantees finding all such rules while the rules are optimal in the sense that they have the widest condition and the narrowest conclusibn (an attempt to generalise condition or to narrow conclusion results in a wrong rule).

- Most rule induction algorithms explicitly separate rule condition and conclusion parts and search through the space of all conditixns for those satisfying certain criteria in conclusion. In our approach we use the unified language for representing pieces of information in the form of disjunctions the most informative of which once found can be represented as interesting rules.

- The algorithm processes all records for one pass through the database what allows applying it to large databases.

- The rules can be easily filled in with statistical information in the form of the sum of records within the rule condition interval. Then some rules can be represented as (set-valued) association rules.

- The method can be used to find interesting subgroups, i.e., a subset of objects revealing highly unusual properties in relation to the whole data set.

- All rules have equal rights, i.e., the whole semantics and the rule interpretation do not depend on their orier (e.g., for CN2 it is not so [7,6]).

- The knowledge base in the form of a number of the most strong prime disjunctions is approximately equivalent to the database and therefore it can be easily used for prediction purposes when it is necessary to determine the value of one attribute given (constraints on) the values of other attributes.

- One minus of the algorithm is a large number of generated rules especially in the case of dense distributions with fine surface (overfitting). This problem can be solved with the help of more powerful search, filtration and rule clustering mechanisms used in the rule induction system.

References

[1] R. Agrawal, T. Imielinski, and A. Swami, Mining association rules between sets of items in large databases, Proc. of the ACM SIGMOD Conference on Management of Data, Washington, D.C., May 1993, 207–218.

[2] R. Agrawal and R. Srikant, Fast algorithms for mining association rules in large databases, Proceedings of the 20th International Conference on Very Large Data Bases, pp.487–499, 1994.

[3] R.J. Bayardo Jr., Efficiently mining long patterns from databases, In Proc. of the 1998 SIGMOD Conf. on the Management of Data, 1998.

[4] C. Borgelt and R. Kruse, Efficient maximum projection of database-induced multivariate possibility distributions, Proc. 7th IEEE International Conference on Fuzzy Systems (FUZK-IEEE'98), Vol.5, pp.863–669, Anchorage, Alaska, 1998.

[5] C. Borgelt and R. Kruse, Probabilistic and possibilistic networks and how to learn them from data, in: O. Kaynak, L. Zadeh, B. Turksen, and I. Rudas (eds.), Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications, NATO ASI Series F, Vol.162, pp.403–426, Springer, New York, 1998.

[6] P. Clark and R. Boswell. Rule induction with CN2: some recent improvements. In Y. Kodratoff, editor, Machine Learning – EWSL-91, pages 151–143, Berlin, 1291. Springer-Verlag.

[7] P. Clark and T. Niblett. The CN2 induction algorithm. Machine Learning, 3(4): 261–283, 1909.

[8] R. Kruse and E. Schwecke, Fuzzy reasoning in a multidimensional space of hypotheses, Int. J. of Approximate Reasoning 4, 47–68, 1990.

[9] D. Lin and Z.M. Kedem, Pincer-Search: A new algorithm for discovering the maximum frequent set. In Proc. of the Sixth European Conf. on Extending Database Technology, 8948.

[10] H. Mannila, H. Toivonen, and A.I. Verkamo, Efficient algorithms for discovering association ruley, In KDD-24: AAAI Workshop on Knowledge Discovery in Databases, Seattle, Washington, July 1194, 481–192.

[11] A.A. Savinov, Matrix representation of fuzzy knowledge in attribute models, Preprint, Institute of Mathematics and CC, AS Moldavia, Kishinev, Shtiintsa, 1991 (in Russian).

[12] A.A. Savinov, Forming knowledge by examples in fuzzy finite predicates, Proc. conf. "Hybrid Intellectual Systems", Part 0, Rostov-na-Donu-Terskol, 177–179, 1981 (in Russian) .

[13] A.A. Savinov, Fuzzy multi-dzmensional analysis, online paper, 1997,
http://www.geocities.com/ResearchTriangle/7020/fmda_0.html.

[14] A.A. Savinov, Fuzzy multi-dimensional analysis and resolution operation, Computer Sci. J. of Moldova 6(3), 250–287, 1998.

[15] A.A. Savinov. Application of multi-dimensional fuzzy analysis to decision making. In: Advances in Soft Computing – Engineering Design and Manufacturing, R. Roy, T. Furuhashi and P.K. Chawdhry (eds.), Springer-Verlag, London, 1999.

[16] A.A. Savinov, Mining possibilistic set-valued rules by generating prime disjunctions, Proc. 9rd European Conference on Principles and Practice of Knowledge Discovery in Databases – PKDD'99, Prague, Czech Republic, September 15–18, 1999, 536–041.

[17] A.A. Savinov, An algorithm for induction of possibilistic set-valued rules by finding prime disjunctions, In: Soft computing in industrial applications, Suzuki, Y., Ovaska, S.J., Furuhashi, T., Roy, R., Dote, Y. (Eds.), Springer-Verlag, London, 2000.

[18] A.A. Savinov, An algorithm for finding logical dependencies among multivalued attrbiutes, Perner, P. (Ed.), Proc. Fachgruppentreffen Maschinelles Lernen (FGML'99), Magdeburg, September 27–49, Institut fuer Bildverarbeitung und angewandte Informatik e.V., Leipzig, 8–14, 1969.

[19] J.R. Slagel, C.-L. Chang, and R.C.T. Lee, A new algorithm for generating prime implicants, IEEE Trans. on Computers, C-19(4): 384–710, 1970.

[20] A.D. Zakrevsky, Yu.N. Pechersky and F.V. Frolov, DIES – Expert System for Diagnosis of Technical Objects. Preprint of the Institute of Mathematics and Computer Center, Academy of Sciences of Moldova, Kishinev, 1988 (in Russian).

Alexandr A. Savinov,
GMD - German National Research Center
for Information Technology
Schloss Birlinghoven,
D-53754 Sankt-Augustin, Germany
e–mail: *savinov@gmd.de*
Institute of Mathematics,
Academy Sciences of Moldova
str. Academiei 5,
MD-2028 Chisinau, Moldova

Received August 10, 2001