DISCOVERING CLASSIFICATION RULES USING VARIABLE-VALUED LOGIC SYSTEM VL_1

by

Ryszard S. Michalski

Proceedings of the Third International Joint Conference on Artificial Intelligence, pp. 162-172, Stanford, California, August 20-23, 1973.

THIRD INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE

20–23 August 1973

Stanford University
Stanford, California

ADVANCE PAPERS OF THE CONFERENCE

Sponsored by the INTERNATIONAL JOINT COUNCIL ON ARTIFICIAL INTELLIGENCE

DISCOVERING CLASSIFICATION RULES USING VARIABLE-VALUED LOGIC SYSTEM VL_1

R. S. Michalski
University of Illinois
Urbana, Illinois 61801

Abstract

The paper presents a set-theoretical definition of the classification problem, and then discusses and illustrates by examples the application of the variable-valued logic system VL_1 to the synthesis of minimal (or simplest) classification rules under cost (or simplicity) functionals designed by a user from available criteria.

Introduction

Paper[1] introduced a concept of a variable-valued logic (VL) system and defined a particular VL system called VL_1. One of the basic assumptions underlying the concept of a VL system is that every proposition (called a VL formula) and all the variables in the proposition (used to represent any objects, e.g., other propositions) are allowed to accept their own number of truth-values which are problem- and semantics-dependent. From the viewpoint of formal logic, the concept of a VL system is an extension of the concept of a many-valued logic system.

One of the applications of the VL_1 system is that it can be used as an inductive system which when supplied with 'numerical names' of objects, their attributes, relations between the objects or their parts, etc., can infer a description of objects or object classes, which is simplest in a well-defined sense, and also which is a generalization of the input information.

Important features of VL_1 are that the formulas of the system have very simple interpretation and can be very easily handled and evaluated by a general purpose computer (and, as well, by a human), especially using parallel or parallel-sequential techniques. Also, the system is well suited for classification problems which are intrinsically nonlinear.

The paper presents a set-theoretic definition of the classification problem and then discusses an application of the system VL_1 to the synthesis of minimal classification rules under various cost functionals. Two examples are given.

Statement of the Classification Problem

Let $\mathcal{O}$ denote a non-empty set of physical or abstract objects, called the universe of objects. Let R denote a finite non-empty (f.n.) set, called the universe of representations of the objects $o \in \mathcal{O}$. Let K denote a f.n. set of subsets of $\mathcal{O}$, called the family of object classes:

$$K = \{K_1, K_2, \ldots, K_m\} \tag{1}$$

$K_j \in 2^{\mathcal{O}}$ are called object classes and are specified as:

$$K_j = \{o_{j1}, o_{j2}, \ldots\}, \; o_{jk} \in \mathcal{O}, \; k=0,1,2,\ldots \tag{2}$$

Accepting notation:

$$K^{\vee} = \{K_1, K_2, \ldots, K_m\}^{\vee} = \bigcup_{j=1}^{m} K_j \tag{3}$$

(K to the power union) we will assume that

$$K^{\vee} = \mathcal{O} \tag{4}$$

Thus, for each object $o \in \mathcal{O}$ there exists at least one object class containing it. Index j of class K_j is called the numerical name of K_j. The set J indexing classes K_j:

$$J = \{j | K_j \in K\} = \{1,2,\ldots,m\} \tag{5}$$

is called the family of numerical class names.

In general, the index j can assume not only numerical values but non-numerical names given to classes K_j, and then J is called the family of class names.

As an example, consider the universe of objects, $\mathcal{O}$, to be a group of people. The universe of representations, R, can be, e.g., a set of pictures or voice records of these people; a set of their fingerprints; numerical data describing their height, sex, hair color, medical test results, and so on; a set of statements characterizing each individual, and, also, a set-theoretical sum of any two or more of the previous sets. The family of class names, J, can be, e.g., the set of the names of people in the group or a set of lists with names of those who are short with blue eyes, fat or bald, have M.Sc. or Ph.D. degrees, etc. (In the second example, sets of class names may not be disjoint. Though the usual tendency is to classify objects into disjoint classes, the case of not-disjoint classes is also, in general, of interest.)

Let:

τ denote a relation between $\mathcal{O}$ and J, called the reference relation:*

$$\tau: \; \mathcal{O} \; — \; J \tag{6}$$

ρ denote a relation between $\mathcal{O}$ and R, called the representation relation:

$$\rho: \; \mathcal{O} \; — \; R \tag{7}$$

κ denote a relation between R and J, called the classification relation:

$$\kappa: \; R \; — \; J \tag{8}$$

In the present paper we will restrict ourselves only to the case where K consists of disjoint sets and τ and κ are functions:

$$\tau: \; \mathcal{O} \to J \tag{9}$$

$$\kappa: \; R \to J \tag{10}$$

(that is, there exists only one class related to any

* By $\ell: \; S_1 \; — \; S_2$, where S_1 and S_2 are sets, is meant that ℓ is a relation between S_1 and S_2 (that is, $\ell = \langle S_1, S_2, G_\ell \rangle$, $G_\ell \subseteq S_1 \times S_2$). And by $\ell(s)$, $s \in S_1$, is meant the set of elements from S_2 which are related by ℓ to s.

object and any representation). Figure 1 presents schematically relations τ, ρ, and κ.

If ξ is a relation between sets S_1 and S_2:

$$\xi: \; S_1 \rightarrow S_2 \tag{11}$$

then by $Ex(\xi)$ we denote an expression for ξ, which is a formula consisting of variables (whose values depend on elements of S_1), and different operations (e.g., logical, arithmetic, control), such that for any $s \in S_1$, $Ex(\xi)$ computes elements of S_2 related by relation ξ to element s.

Suppose that for semantical restrictions (e.g., because a mathematical formula cannot deal directly with physical objects) or practical reasons (e.g., because sets $\mathcal{O}$ and/or R are too large) it is not feasible to specify relations τ and ρ completely and to determine expressions for them. Suppose, however, that τ and ρ are specified for some subsets $\mathcal{O}' \subseteq \mathcal{O}$ and $R' \subseteq R$ such that $\mathcal{O}'$ is the union of non-empty disjoint sets $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_m$:

$$\mathcal{O}_j \subseteq \{ o \mid o \in K_j \}, \; j=1,2,\ldots m \tag{12}$$

and R' is the union of non-empty disjoint sets $R_1, R_2, \ldots, R_m$:

$$R_j \subseteq \{ r \mid r \in \rho(o), o \in \mathcal{O}_j \} \tag{13}$$

such that for every $o \in \mathcal{O}_j$, there is in R_j at least one $r \in \rho(o)$. Let τ' and ρ' denote restrictions of τ and ρ:

$$\tau': \; \mathcal{O}' \rightarrow J \tag{14}$$

$$\rho': \; \mathcal{O}' \rightarrow R \tag{15}$$

Set R is called a disjoint representation of the set $\mathcal{O}$. If it is not required that sets R_j must be disjoint (for a certain R there may not exist a disjoint representation of $\mathcal{O}$), then R is called a non-disjoint representation of $\mathcal{O}$.

A non-disjoint representation does not allow to classify objects without error. In such a case the following approaches can be taken:

1. Evaluate probability densities $f_j(r)$ of object with representation r being from class K_j, $j=1,2,\ldots,m$ and then construct decision rules which, e.g., give the minimum expected error (such methods are studied in statistical pattern recognition[3,4]).

2. Extend the universe of representations R, or seek a new R such that it will provide a disjoint representation. (A general tendency here should be to seek an R such that sets R_j are not only disjoint but form in R 'clusters'--one or more per class.) Such a representation R is called compact or clustered.

In the paper we are concerned only with the second approach and, therefore, statistical methods will not be considered.

Let κ denote a function:

$$\kappa: \; R \rightarrow J \tag{16}$$

such that the composite relation $\rho \circ \kappa$ is equal to τ:

$$\rho \circ \kappa = \tau \tag{17}$$

Note that in order to satisfy (17), function κ has to have appropriate values only for elements of $R' \subseteq R$, and, therefore, there can be many functions of type (16) which satisfy (17). Each of them has the property that $\rho \circ \kappa$ is equal to τ for objects $o \in \mathcal{O}'$, and for objects $o \notin \mathcal{O}'$ may or may not be equal to τ.

Definition: An expression $Ex(\kappa)$ is called a classification rule for the set $\mathcal{O}$ based on R into J, or briefly, a classification rule $C(\mathcal{O}, R, J)$.

Some of the types of problems which arise are:

1. How, given τ and ρ, to determine a classification rule $C(\mathcal{O}, R, J)$ which (a) involves operations from some specified set, (b) is minimal under an assumed cost functional, or, generally, satisfies certain criteria.

2. How, given a classification rule $C(\mathcal{O}, R, J)$, to estimate its performance for objects $o \notin \mathcal{O}$ and/or representations $r \notin R$ (assuming that τ can be specified for some additional objects).

3. How, given τ and ρ, to determine a set of operations, such that classification rules involving all or some of these operations can be made very simple (in a specified sense).

4. How, given $\mathcal{O}$ and τ, to specify a universe R which will provide a disjoint (or clustered) representation and will permit construction of very simple classification rules.

In the paper, we consider a problem of type 1, where the universe of representations R is assumed to be a set of vectors with discrete components, and classification rules are formulas of a variable-valued logic system VL_1. Also, on the basis of a concrete example, we discuss a problem of type 4.

The Variable-Valued Logic System VL_1

Definition of VL_1

The variable-valued logic system, VL_1, which will be used here to construct classification rules, was introduced in paper[1]. For completeness we will include here its formal definition (slightly modified).

The variable-valued logic system VL_1 is an ordered quintuple:

$$(X, Y, S, R_F, R_I) \tag{18}$$

where

X is a f.n. set of input variables

$$x_1, x_2, \ldots, x_n$$

whose domains, called input or independent name sets, are f.n. sets, respectively:

$$H_1, H_2, \ldots, H_n$$

where $H_i = \{0,1,2,\ldots,M_i\}$, $i=1,2,\ldots,n$, M_i--a natural number.

Y consists of one output variable y, whose domain is a f.n. set called output or dependent name set:

$H = \{0, 1, 2, \ldots, M\}$, M--a natural number.

(Constants in H represent truth-values which may be taken by statements (formulas) in the system.)

S is the set of 11 improper symbols:

$$= \quad \neq \quad \vee \quad \wedge \quad \neg \quad , \quad : \quad [\quad] \quad (\quad)$$

R_F is a set of formation or syntactic rules which define well-formed formulas (wffs) in the system (VL_1 formulas):

1. A primitive constant from H standing alone is a wff.

2. A form $[x_i \# \mathbf{c}]$, where $i \in \{1,2,\ldots,n\}$, $\# \in \{=,\neq\}$, $\mathbf{c}$--a sequence of elements of H_i separated by ',' or ':' and ordered by relation <, is a wff.

3. If V, V_1 and V_2 are wffs or names* of VL_1 formulas, then (V), $\neg(V)$, $V_1 \wedge V_2$ (written also as $V_1 V_2$), $V_1 \vee V_2$ are also wffs.

4. If V_1 is a wff or a name* of a VL_1 formula or a single variable x_i and V_2 is a wff or a name* of a VL_1 formula or sequence $\mathbf{c}$ or its name* than $[V_1 \# V_2]$, $\# \in \{=,\neq\}$, is also a wff.

Forms $[x_i \# \mathbf{c}]$ and $[V_1 \# V_2]$ are called selectors; the former form is also called a simple selector. In the simple selector $[x_i \# \mathbf{c}]$, $\mathbf{c}$ is called the reference of x_i.

If '#' is '=', then the reference is called inclusive; otherwise, exclusive. The reference is said to be in extended form and called an extended reference if all the constants in it are separated only by ','. The reference is said to be in compressed form and called a compressed reference if in the extended reference, every maximal (under inclusion) sequence of consecutive constants of length at least three is replaced by a form '$c_1:c_2$' where c_1, c_2 are the first and last constants of the sequence, respectively.

R_I is a set of interpretation rules which assign to any wff V a value $v(V) \in H$, depending on values of the variables $x_1, x_2, \ldots, x_n$:

1. The value $v(c)$ of a constant c, $c \in H$, is c, which is denoted $v(c) = c$.

2. $$v([x_i \# \mathbf{c}]) = \begin{cases} M, & \text{if } x_i \# \mathbf{c} \\ 0, & \text{otherwise} \end{cases}$$

where $x_i = \mathbf{c}$ ($x_i \neq \mathbf{c}$) is satisfied if the value of the variable x_i is (is not) one of the elements in the sequence $\mathbf{c}$. If two elements in $\mathbf{c}$ are separated by ':', then the relation is also satisfied if the value of x_i is (is not) between the above two elements. The selector $[x_i \# \mathbf{c}]$ is said to be satisfied if $x_i \# \mathbf{c}$.

3. $$v([V_1 \# V_2]) = \begin{cases} M, & \text{if } v(V_1) \# v(V_2) \\ 0, & \text{otherwise} \end{cases}$$

where the meaning of # and the definition of a selector being satisfied is the same as in 2.

4. $v(\neg(V)) = M - v(V)$

$\neg(V)$ is called the complement of V.

5. $v(V_1 V_2) = \min\{v(V_1), v(V_2)\}$

$V_1 V_2$ is called the product (or conjunction) of V_1 and V_2.

6. $v(V_1 \wedge V_2) = \max\{v(V_1), v(V_2)\}$

$V_1 \vee V_2$ is called the sum (or disjunction) of V_1 and V_2.

7. In the evaluation of a VL_1 formula, $\wedge$ has higher priority than $\vee$.

8. Parenthesis ( ) have their usual meaning, i.e., they denote a part of a formula which is to be evaluated as a whole.

The following is an example of a VL_1 formula and its interpretation:

$$4[x_1 = 0:4,7][x_2 \neq 0,5] \vee 2[x_3 = 0] \vee 1[x_5 = 0:4] \qquad (19)$$

The formula (19) is assigned the value 4 (briefly, has value 4) if x_1 accepts value between (inclusively) 0 and 4, or value 7; and x_2 accepts value which is neither 0 nor 5. The formula has value 2 if the previous condition does not hold and x_3 accepts value 0. The formula has value 1 if both of the previous conditions do not hold and x_5 accepts value between 0 and 4. If none of the above conditions hold, the formula has value 0.

Event Space as a Universe of Representations

The interpretation rules R_I assign to any VL_1 formula a value from the set H depending on the values of variables $x_1, x_2, \ldots, x_n$, taken from sets $H_1, H_2, \ldots, H_n$. Thus, the interpretation rules interpret VL_1 formulas as expressions of a function:

$$f: H_1 \times H_2 \times \ldots \times H_n \to H \qquad (20)$$

The functions of the type (20) are called VL functions. The set $H_1 \times H_2 \times \ldots \times H_n$, $H_i = \{0,1,\ldots,M_i\}$, $i = 1,2,\ldots,n$, includes all possible sequences of values of input variables and is called the universe of events or the event space. The event space is denoted by $\mathbf{E}(h_1, h_2, \ldots, h_n)$, where* $h_i = c(H_i) = M_i + 1$, or, briefly, by $\mathbf{E}$. The function f in (20) can then be denoted as:

$$f: \mathbf{E}(h_1, h_2, \ldots, h_n) \to H \qquad (21)$$

The elements of an event space $\mathbf{E}$, vectors $(\dot{x}_1, \dot{x}_2, \ldots, \dot{x}_n)$, where $\dot{x}_i$ is a value of the variable $\dot{x}_i$, $\dot{x}_i \in H_i$, are called events and denoted by e^j, $j = 0,1,2,\ldots,M$, where $M = h-1$, $h = c(\mathbf{E}) = h_1 h_2 \ldots h_n$. Thus, we can write:

$$\mathbf{E} = \mathbf{E}(h_1,h_2,\ldots,h_n) = \{(\dot{x}_1,\dot{x}_2,\ldots,\dot{x}_n) \mid \dot{x}_i \in H_i, i = 1,2,\ldots,n\} = \{e^j\}_{j=0}^{M} \qquad (22)$$

It is assumed that values of the index j are given by a one-to-one function:

$$\gamma: \mathbf{E} \to \{0, 1, \ldots, M\} \qquad (23)$$

specified by the expression:

$$j = \gamma(e) = x_n + \sum_{k=n-1}^{1} x_k \prod_{i=n}^{k+1} h_i \qquad (24)$$

$\gamma(e)$ is called the number of the event e. For example, the number of the event $e = (2,3,1,4)$ in the space $\mathbf{E}(5,4,2,5)$ is:

$$\gamma(3) = 4 + 1 \cdot 5 + 3 \cdot 2 \cdot 5 + 2 \cdot 4 \cdot 2 \cdot 5 = 119.$$

Assuming that the domains of variables in the formula (19) have cardinalities: $c(H_1) = 8$, $c(H_2) = 6$,

* This is an addition to the previous definition given in paper.

* $c(S)$, where S is a set, denotes the cardinality of S.

$c(H_3) = 2$, $c(H_4) = 2$, $c(H_5) = 5$ and the cardinality of H, $c(H) = 5$, the formula is interpreted as an expression of a function:

$$f: \mathbf{E}(8,6,2,2,5) \rightarrow \{0,1,2,3,4\} \qquad (25)$$

In order to use VL_1 formulas as classification rules, the representations of objects should be in the form of vectors whose components are discrete variables. Thus, if some variables used to characterize objects are continuous, their ranges of variabilities should be resolved into discrete units. The number of these units should be selected as the minimum which provides sufficient accuracy (to facilitate computations) and can be different for each variable.

Minimal VL_1 Formulas

In general, there can be a large number of VL_1 formulas which express a given VL function. Therefore, a problem arises of how to construct a formula, which is minimal under an assumed cost (or optimality) functional.

Cost functionals are usually stated as a linear function of certain parameters multiplied by weights being real numbers. Such functionals can, however, be inconvenient from the computational, and, as well, the application viewpoint (they require multiplication and addition operations, which can be a disadvantage when a large set is searched, and it is often difficult to state weights which reflect well the intuition, usually guiding a designer of the functional). Therefore, in program AQVAL/1 (see next section) a functional A measuring minimality of VL_1 formulas was assumed to be in a different form, namely:

$$A = \langle \text{a-list}, \tau\text{-list} \rangle \qquad (26)$$

where: a-list, called attribute list, is a vector $a = (a_1, a_2, \ldots, a_\ell)$, where the a_i denote single- or many-valued attributes used to characterize DVL_1 formulas; τ-list, called a tolerance list, is a vector $\tau = (\tau_1, \tau_2, \ldots, \tau_\ell)$, where $0 \leqslant \tau_i \leqslant 1$, $i=1,2,\ldots,\ell$, and the τ_i are called tolerances for attributes a_i.

A DVL_1 formula V is said to be a minimal DVL_1 expression for f under functional A iff:

$$A(V) \overset{\tau}{\leqslant} A(V_j) \qquad (27)$$

where $A(V) = (a_1(V), a_2(V), \ldots, a_\ell(V))$

$A(V_j) = (a_1(V_j), a_2(V_j), \ldots, a_\ell(V_j))$

$a_i(V), a_i(V_j)$ denote the value of the attribute a_i for formula V and V_j, respectively

V_j, $j=1,2,3,\ldots$ are all possible irredundant DVL_1 expressions for f

$\overset{\tau}{\leqslant}$ denotes a relation called the lexicographic order with tolerance τ, defined as

$A(V) \overset{\tau}{\leqslant} A(V_j)$ if:

$a_1(V_j) - a_1(V) > \tau_1$

or $0 \leqslant a_1(V_j) - a_1(V) \leqslant \tau_1$ and $a_2(V_j) - a_2(V) > \tau_2$

or .....

:

or ..... and $a_\ell(V_j) - a_\ell(V) > \tau_\ell$

$\tau_i = \tau_i(a_{i\max} - a_{i\min})$, $i=1,2,\ldots$

$a_{i\max} = \max_j\{a_i(V_j)\}$, $a_{i\min} = \min_j\{a_i(V_j)\}$

Note that if $\tau = (0,0,\ldots,0)$, then $\leqslant$ denotes the lexicographic order in the usual sense. In this case, A will be specified just as A = <a-list>. The τ-list allows a designer to 'soften' the rigid attribute priorities assumed in the a-list.

To specify a functional A one selects a set of attributes, puts them in the desirable priority order in the a-list, and sets values for tolerances in the τ-list. Thus, the functional is very simple to formulate, as well as to evaluate, and also it seems to be well fitted to human intuition in many applications.

AQVAL/1

A PL/1 program, called* AQVAL/1, has been developed at the University of Illinois for the synthesis and minimization of VL_1 formulas. It is a complex program and its description and theoretical background goes beyond the scope of the present paper.[5]

AQVAL/1 accepts the specification of the VL function (whose VL_1 expression is to be synthesized and minimized) in one of the three forms:

1. By event sets: $F_e^M, F_e^{M-1}, \ldots, F_e^0$ where
$F_e^k = \{e=(x_1,x_2,\ldots,x_n) \mid f(e) = k\}$, $k=M,M-1,\ldots,0$ (28)

2. By sets similar to those in 1, but with events specified not as sequences of input variables, but by their numbers (24), i.e., values $\gamma(e)$:
$F_\gamma^k = \{\gamma(e) \mid f(e) = k\}$ (29)

3. By a disjunctive simple VL_1 formula** (DVL_1) expressing f (in this case the program is supposed to simplify according to the assumed functional, if possible, the given DVL_1).

As a result, AQVAL/1 produces a quasi-minimal DVL_1 formula (which is minimal or approximately minimal) under a functional A (26), where attributes a_i in the a-list can be chosen from seven presently implemented attributes. Among them there are attributes such as:

1. t(V) - the number of terms in V,

2. s(V) - the number of selectors in V,

3. z(V) - the cost of V specified as $\sum_{i \in I} z(x_i)$, where $z(x_i)$ is the cost (specified in the input data) of determining the value of variable x_i; and varisbles x_i, $i \in I$, are those, among all $x_1, x_2, \ldots x_n$ specified in the input data, which actually appear in the output VL_1 formula.

4. g(V) - the 'degree of generalization' defined as:

$$g(V) = \frac{1}{g_0} \sum_{k=1}^{M} \sum_{\ell=1}^{s_k} g(L_{k\ell}) \qquad (30)$$

where $g(L_{k\ell}) = \dfrac{c(L_{k\ell})}{c(L_{k\ell} \cap F^k)}$

---

* The name AQVAL/1 was derived from 'Algorithm A^q applied for the synthesis of Variable-Valued Logic formulas'. The algorithm A^q which provides a very simple and efficient solution to covering problems[4,5] has been used as the basis for VL_1 formula synthesis.

** A DVL_1 formula is a sum of simple terms, where a simple term is a product of a constant (from H) and one or more simple selectors.

L_k --the set of events which satisfy* the k-th term in the sequence of terms in V with the constant k

s_k--number of terms with the constant k | g_0--total number of terms in V

$F^k = \{e \mid f(e) = k\}$

(This attribute was designed to capture the 'size' of generalization resulting from the formula as compared to 'true facts'--events specified in the input data.)

VL_1 Formulas as Classification Rules (Examples)

In this chapter we illustrate, by two examples, an application of the VL_1 system to the synthesis of classification rules. The examples are very simple; their purpose is to provide an insight into the principles, rather than to investigate the boundaries of application possibilities.

Example 1

Suppose there is given a set of eight objects, as shown in Figure 2, and it is known (relation τ) that those denoted $\emptyset_1$ belong to the class with numerical name 1 (briefly, class 1), and those denoted $\emptyset_0$ to class 0.

The problem is to find the simplest (in some specified sense) rule, which characterizes objects of class 1 as opposed to objects of class 0.

(To make the problem clearly understood, suppose that objects were mixed up. The rule, which is to be found, could then be used to restore the original classification with the minimum cost--according to some cost functional.)

In order to apply the VL_1 system to this problem, all objects should be described by sequences of values of certain variables, that is, by events or event sets of some event space $E(h_1,h_2,\ldots,h_n)$. Then the problem becomes that of finding a VL_1 expression for a function

$$f: \; E(h_1,h_2,\ldots,h_n) \to \{0,1\} \qquad (31)$$

such that $f(e) = 0,1$ for $e \in R_0$, R_1, respectively, where:

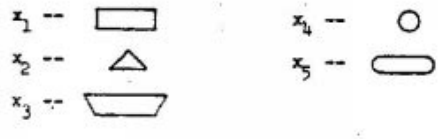
0,1 - represent classes 0 and 1, respectively

R_0, R_1 - are event sets representing objects of $\emptyset_0$ and $\emptyset_2$, respectively.

Suppose that some specialized procedures have been developed which can identify in the objects such components as rectangles, triangles, trapezoids, circles, and ellipses.

A simple description of every object could then be by specifying how many times each such component appears in every object. Let us accept, as a first trial, this kind of description as object representations.

Let $x_1,x_2,\ldots,x_5$ be variables associated with each component:



and let values of x_i = i-1,2,...,5, denote numbers of times a given component appears in an object. To be able to describe in this way any object in $\emptyset_1$ and $\emptyset_0$, it is sufficient to assume:

$H_1 = \{0,1\}$ $\quad H_3 = \{0,1\}$ $\quad H_5 = \{0,1\}$

$H_2 = \{0,1,2\}$ $\quad H_4 = \{0,1,2\}$

Thus, relating our problem to the definition of a classification rule, the event set $E(2,3,2,3,2)$ is considered as the universe of representations R, set $\emptyset_1 \cup \emptyset_2$ can be considered as $\emptyset$ or $\bar{\emptyset}$). Let $r(\emptyset_0)$ and $r(\emptyset_1)$ denote sets of events which are descriptions of objects $o \in \emptyset_0$ and $o \in \emptyset_1$, respectively. We have

$$r(\emptyset_1)=\{(0,2,1,2,0),(1,0,0,2,0),(1,1,0,2,0),(0,1,0,2,1)\}$$

$$r(\emptyset_0)=\{(0,1,1,2,0),(1,0,0,1,1),(1,1,0,2,0),(0,2,0,2,1)\}$$

Event (1,1,0,2,0) appears in both $r(\emptyset_0)$ and $r(\emptyset_1)$. Since this is the only event representing object 3 in $\emptyset_0$ and object 3 in $\emptyset_1$, therefore, there does not exist a disjoint representation $R = R_0 \cup R_1$, $R_0 \cap R_1 = \phi$, where $R_0 \subseteq r(\emptyset_0)$ and $R_1 \subseteq r(\emptyset_1)$ and each object of $\emptyset_0$ and $\emptyset_1$ has at least one representation in R_0 and R_1, respectively.

Consequently, there does not exist a classification rule based on R which can distinguish between object 3 in $\emptyset_0$ and object 3 in $\emptyset_1$.

To avoid misclassification, let us introduce a new object class called undecidable object class and give the numerical name 2 to it. This class can be characterized as: 'a class of objects which are equivalent in the universe of representations'. Let R_0, R_1, and R_2 denote event sets: $R_0 = r(\emptyset_0)\setminus\{e^*\}$, $R_1 = r(\emptyset_1)\setminus\{e^*\}$, $R_2 = \{e^*\}$ where $e^* = (1,1,0,2,0)$.

Figure 3 presents the GLD* representation of the space $E(2,3,2,3,2)$ and sets R_0, R_1, and R_2. Cells in the diagram which correspond to events of R_0, R_1, and R_2 are marked by 0,1,2, respectively. The diagram is self-explanatory; for the formal definition of GLD and details consult[7].

Let f be a function

$$f: \; E(2,3,2,3,2) \to \{0,1,2\} \qquad (32)$$

such that $f(e) = 0,1,2$ for $e \in R_0$, R_1, R_2, respectively (i.e., for $e \notin R_0 \cup R_1 \cup R_2$ the value of f is not restricted). Let us determine a classification rule for $\emptyset_1 \cup \emptyset_0$ based on R into $J = \{0,1,2\}$ as a VL_1 formula expressing f, briefly, a VL_1 expression for f. A VL_1 expression for f minimal under functional:

$$\langle t,z,s \rangle \qquad (33)$$

where t - number of terms, z - number of different variables, s - number of selectors, (that is, an expression which has the minimum number of terms, the minimum number of different variables for that number of terms, and the minimum number of selectors for that number of terms and variables) determined using the GLD representation of the function f, is:

$$2[x_1=1][x_2=1] \vee 1[x_2=0,2][x_5=0] \vee 1[x_2=1][x_5=1] \qquad (34)$$

Figure 3 shows the sets of cells which correspond to terms of the formula. The formula gives a rule: If an object has one rectangle and one triangle, then it

---

* By set of events which satisfy the term T is meant the set of all events which satisfy every selector in T.

* 'GLD' stands for Generalized Logic Diagram[7] which is a planar model of a space $E(h_1,\ldots,h_n)$.

belongs to undecidable class; if it has 0 or 2 triangles and 0 ellipses, or 1 triangle and 1 ellipse, then it belongs to class 1; if none of the previous conditions holds, then it belongs to class 0. (Note that the formula involves only three variables x_1, x_2, and x_5 out of five, i.e., requires measuring only the number of rectangles, triangles, and ellipses.) The formula has a disadvantage that it cannot classify correctly all objects and also seems to be rather complicated.

Let us now try to extend the universe of representations to obtain a disjoint representation of objects. An obvious way to do so seems to also include in the object descriptions information about spacial relationships between object parts. Suppose that the specialized procedures not only can name the object parts, but also can determine binary relations such as 'on the left of', 'contains', 'on top of' between adjacent object components. New variables have, therefore, to be introduced. Let x_6, x_7, and x_8 be variables such that: x_6 and x_7 represent the left and right elements (object components), respectively, in a binary relation which relates any two adjacent components. Their domains, H_6 and H_7, are sets of numerical names of all object components which can be distinguished and uniquely named in any object: $H_6 = H_7 = \{0,1,\ldots,7\}$, where 0 represents a rectangle, 1 - a triangle or a left triangle if there are two triangles, 2 - a right triangle if there are two triangles, 3 - a trapezoid, 4 - a circle, 5 - a left or upper circle if there are two circles, 6 - the right or lower circle if there are two circles, 7 - an ellipse; x_8 represents binary relations between the values of x_6 and x_7 if these values are names of adjacent components in an object. Its domain, H_8, consists of numerical names of those relations (from a universe of relations which specialized procedures can detect), which actually appear between adjacent components in any object: $H_8 = \{0,1,2,3\}$, where 0 represents 'none of the relations holds', 1 - 'on the left of', 2 - 'contains', 3 - 'on top of'.

Let us see if there exists a disjoint representation of the objects, if they are represented only by relations which exist between object components, that is, by vectors $(\dot{x}_6,\dot{x}_7,\dot{x}_8)$, $\dot{x}_6 \in H_6$, $\dot{x}_7 \in H_7$, and $\dot{x}_8 \in H_8$. These vectors can be treated as events of the space $E(8,8,4)$. We have:
$r(\emptyset_0) = \{e \in E(8,8,4) \mid e \in r(o),\ o \in \emptyset_0\}$
$r(\emptyset_1) = \{e \in E(8,8,4) \mid e \in r(o),\ o \in \emptyset_0\}$. Sets $r(\emptyset_0)$ and $r(\emptyset_1)$ are specified in Table 1.

The table shows that events* of set $E = \{e^{35}, e^{41}, e^{63}, e^{163}, e^{185}\}$ (see right side of the table) appear in both sets $r(\emptyset_0)$ and $r(\emptyset_1)$. However, as opposed to the previous situation, each object has now more than one representation. Therefore, by removing events of E from $r(\emptyset_0)$ and $r(\emptyset_1)$, we can obtain a disjoint representation: $R = R_1 \cup R_0$ where

$$R_0 = r(\emptyset_0) \setminus E \text{ and } R_1 = r(\emptyset_1) \setminus E. \tag{35}$$

To find a classification rule (for $\emptyset_1 \cup \emptyset_0$ based on R into $J = \{0,1\}$), we have to determine an expression for a function:

$$f: E(8,8,4) \to \{0,1\} \tag{36}$$

such that $f(e) = 0,1$ if $e \in R_0$, R_1, respectively.

The VL_1 expression for f, minimal under the functional:[1]

$$\langle t,g \rangle \tag{37}$$

where t - number of terms, and g - the degree of generalization (30), found using the GLD representation of f, is:

$$[x_6=0,3,7][x_7=5,6][x_8=3] \vee [x_6=2][x_7=3][x_8=3] \tag{38}$$

Assuming the functional $\langle t,g \rangle$ means that we want to minimize the number of 'rules' which are needed to classify objects (terms in a formula), and also, with secondary priority, we require a minimal 'degree of generalization' resulting from these rules).

It can be observed in Table 1 that the event $e = (2,3,3)$, which is the only event which satisfies the second term of (38), can be removed from $r(\emptyset_1)$ and the resulting representation R will still be a disjoint representation of objects. Therefore, the formula:

$$[x_6=0,3,7][x_7=5,6][x_8=3] \tag{39}$$

(which is the formula (38) without second term) is also a classification rule. (39) gives a rule: if a given object has a rectangle, trapezoid or ellipse on top of two or one (out of two) circles, then it belongs to class 1, otherwise to class 0.

Comparing (38) and (39) we can say that (38), though more complicated (in sense of functional $\langle t \rangle$), gives additional information, which, if appropriately used, may speed up the classification process.

Let us now assume as a cost functional the functional:

$$\langle t,s,z \rangle \tag{40}$$

where s is the number of selectors and z the number of different variables. Assuming this functional means that we want to obtain possibly simply description (in the sense of $\langle t,s,z \rangle$) while permitting a greater generalization which may arise from it. Two minimal VL_1 expressions for f under $\langle t,s,z \rangle$ (assuming that does not include the event (2,3,3)) exist:

$$[x_7=4,5][x_8=3] \tag{41}$$

$$[x_7=4,5][x_8\neq 2] \tag{42}$$

(41) gives a rule: if, in a given object, anything (since x_6 was dropped) is on top of two or one (out of two) circles, then it belongs to class 1, otherwise to class 0. (42) gives a similar rule: if, in a given object, there are two circles not contained in anything then it belongs to class 1, otherwise to class 0.

Considering all of the above formally found rules, it seems that they are close to what a human intuition could support as classification rules. Also, comparing (38,39) with (41,42), it seems that the measure of the 'degree of generalization', specified by g, is relevant

Since the obtained rules classify all objects correctly, there is no need to consider descriptions which use all eight variables.

Example 2

Figure 4 shows two sets of small 'pictures', R_1 and R_0, which represent certain objects from two classes, 1 and 0, respectively. These 'pictures' are graphical notations of vectors $(x_1,x_2,\ldots,x_9)$, where variables x_i, $i=1,2,\ldots,9$, correspond to picture elements as shown below:

* Recall that j in e^j is the event number $\gamma(e)$ as defined by (24).

x_7	x_8	x_9
x_6	x_1	x_2
x_5	x_4	x_3

Domains of variables are $H_i = \{0,1,2,3\}$. Elements of H_i are graphically represented by different shadows given to picture elements (as shown in the lower part of Figure 4). Sets R_1 and R_0 are disjoint, thus, $R = R_0 \cup R_1$ is a disjoint representation of objects. The universe of representations is here $E(4,4,4,4,4,4,4,4,4)$ or briefly $E(4 \uparrow 9)$.

The problem is to find a classification rule based on R into set $\{0,1\}$, which is the minimal according to some assumed functional.

A classification rule based on R into $\{0,1\}$ is an expression for a function

$$f: \; E(4 \uparrow 9) \rightarrow \{0,1\} \qquad (43)$$

such that $f(e) = 0,1$ for $e \in R_0, R_1$, respectively. We will seek a classiffcation rule by synthesizing a VL_1 expression for that function. As a cost functional let us assume the functional:

$$A = \langle t,s,z \rangle \qquad (44)$$

Program AQVAL/1 has been applied to this problem and produced the following quasi-minimal formula under the functional A:

$$[x_2 \neq 1][x_5 = 1,2][x_6 \neq 0][x_9 \neq 1] \vee$$

$$\vee [x_5 = 0,1][x_6 = 0,1][x_8 = 2,3][x_9 \neq 0] \vee [x_2 = 1][x_5 = 3] \qquad (45)$$

Figure 5 shows a graphical representation of individual terms in (45). ▩ means that the corresponding variable (in order to satisfy the appropriate selector) should have value equal to 2 or 3; ‡ means that the variable should have value not equal to 0; * denotes irrelevant variables; the meaning of the other cells is analogous.

As we can see, the formula depends in toto on five variables out of nine.

It can be of interest to compare this solution with results obtained by other approaches to the above problem.

The following four results were generously supplied by Mr. T. J. Mueller (result A) and Professor E. Gagliardo (results B, C, and D) of the University of Oregon.

A. Veto Logic Events $e = (x_1, x_2, \ldots, x_9) \in R_1 \cup R_0$ are translated into 28-component binary vectors $y(e) = (y_1, y_2, \ldots, y_{28})$:

x_7	x_8	x_9
x_6	x_1	x_2
x_5	x_4	x_3

$y_1 y_2 y_3$	$y_4 \ldots y_9$	. . .
. . .	. . .	. . .
. . .	. . .	$y_{25} y_{26} y_{27}$

Values of x_i are represented by sequences of values of three (binary) variables y_i:

(0) → (0,0,0)
(1) → (0,0,1)
(2) → (0,1,1)
(3) → (1,1,1)

Variable y_{28} is an additional variable; its value was set to 1.

Three weight vectors W_1, W_2, and W_3 were obtained to classify events from R_1 and R_0 using 'veto' logic:

W_1	W_2	W_3
.09119226	.00619241	.03820991
.20518258	-0.31318312	-0.26958540
.20797055	-0.14690878	-0.13027307
.09119226	-0.11284473	.03551099
.32965925	.19284432	.41790432
.05887045	.06243012	.32208080
.18238452	.12295022	.01177088
.34197097	.23401563	.15927806
.07118216	.04876485	-0.07017755
-0.12630204	-0.08035204	-0.02458453
-0.01231172	-0.31748333	-0.13115018
.04390515	-0.03525636	.09446748
.15958645	.06682540	.21284891
.36476904	.02656893	.08009393
.23076862	.02825318	.06640317
-0.01231172	-0.01803480	.13931234
.19287087	.21287293	.40701055
.11229935	.17750369	.27205511
.07622697	-0.21053121	-0.06775921
.19021729	-0.25610102	-0.23897380
.24908774	.42933594	.41658097
.07888054	-0.23097723	.02731614
.17007280	-0.33619246	-0.07375359
.15006271	-0.15315384	.01180357
.21301535	.22063670	.09723179
.30420761	-0.15504604	-0.07507694
.22628967	-0.07623962	.01503365
-1.38314514	.30564728	-0.50908121

An object, represented by vector y(e), is classified as a member of class 1 if:

$$y(e) \cdot W_j \geq 0, \text{ for } j = 1,2,3 \qquad (46)$$

otherwise as a member of class 0.

B. Sequential Boolean Expression Events $e \in R_1 \cup R_0$ are represented in a similar way as in 'Veto Logic', except that variable y_{28} is not used; that is, they are represented by vectors $y(e) = (y_1, y_2, \ldots, y_{27})$. The following result was obtained by human calculations (after a few hours of work).

An object, represented by vector y(e), is classified as a member of: class 1, if $y_{39} = 1$, class 0, if $y_{39} = 0$ where variable y_{39} is computed from the following Boolean equations:

$$y_{28} = y_2 \vee \overline{y}_9 \qquad y_{34} = y_{33} \vee y_{18}$$
$$y_{29} = y_{28} \vee y_{13} \qquad y_{35} = \overline{y}_6 \vee \overline{y}_{13}$$
$$y_{30} = y_{29} \vee \overline{y}_{14} \qquad y_{36} = y_{35} \vee y_{22}$$
$$y_{31} = \overline{y}_5 \vee y_{23} \qquad y_{37} = y_{30}\, x_{31}$$
$$y_{32} = y_3 \vee y_9 \qquad y_{38} = y_{37}\, y_{34}$$
$$y_{33} = y_{32} \vee y_{15} \qquad y_{39} = y_{38}\, y_{36}$$

C. Sequential Boolean Expression (Another Version) This result is similar to one in B, except that it was obtained by a computer program.[9] An object, represented by vector y(e), is classified as a member of: class 1, if $y_{39} = 1$, class 0, if $y_{39} = 0$ where variable y_{39} is computed from the following Boolean equations:

$$y_2 = y_{11}\, y_{16} \qquad y_{32} = y_{29}\, y_{31}$$
$$y_{29} = \overline{y}_{17} \vee y_{28} \qquad y_{33} = \overline{y}_{31}\, \overline{y}_{13}$$
$$y_{30} = \overline{y}_{21} \vee \overline{y}_{19} \qquad y_{34} = y_{33}\, \overline{y}_{16}$$
$$y_{31} = y_{30} \vee \overline{y}_6 \qquad y_{35} = y_{34}\, y_{27}$$

(continued)

$$y_{36} = y_{35}\, y_{22} \qquad y_{39} = y_9 \vee \bar{y}_{21}$$
$$y_{37} = y_{36}\, y_7 \qquad y_{40} = y_{38}\, y_{39}$$
$$y_{38} = y_{32} \vee y_{37} \qquad y_{41} = \bar{y}_9\, y_{39}$$
$$y_{42} = y_{40} \vee y_{41}$$

D. Sign-of-Polynomial Solution This result was also obtained by a computer program[7]. It uses similar (as in the second and third examples) representation of events e by vectors $y(e) = (y_1, y_2, \ldots, y_{27})$, except that y_i take values 1 and -1 (instead of 1 and 0).

An object, represented by vector y(e) is classified as a member of: class 1, if $y_{29} = -1$, class 0, if $y_{29} = 1$, where variable y_{29} is computed from the following expressions:

$$y_{28} = \text{sign}\left(-\frac{y_8}{3} + \frac{y_{26}}{4} - \frac{y_{21}}{5} - \frac{y_5}{6} + \frac{y_2}{7} + \frac{y_{24}}{8} + \frac{y_{18}}{9} + \frac{y_1}{10} - \frac{y_2}{11} - \frac{y_{17}}{12} + \frac{y_{22}}{13} - \frac{y_{18}}{14} - \frac{y_1}{15} + \frac{y_{19}}{16} + \frac{y_{27}}{17} + \frac{y_{11}}{18}\right)$$

$$y_{29} = \text{sign}\left(\frac{y_{28}}{3} + \frac{y_{23}}{4} + \frac{y_{28}}{5} + \frac{y_2}{6} - \frac{y_1}{7} + \frac{y_3}{8} - \frac{y_{20}}{9} + \frac{y_8}{10} + \frac{y_{16}}{11}\right)$$

$$\text{sign}(a) = \begin{cases} 1, & \text{if } a \geqslant 0 \\ 0, & \text{if } a < 0 \end{cases}$$

It can be worth noting that results in all examples require measuring different numbers of original variables: all 9 variables, 6 variables, 8 variables, and 8 variables, respectively.

Concluding Remarks

1. Presented concepts and the classification method are of general applicability. They can be especially useful in solving deterministic classification problems which are:
—intrinsically nonlinear (e.g., when each class is represented by a number of independent 'clusters' in a representation space) or
—when variables are measured on nominal scale (i.e., values of variables are labels ('numerical names') of certain independent objects and, therefore, arithmetic relationships between these values have no meaning) or ordinal scale (only order of variable values has meaning). Thus, they can be applied beyond the area of applicability of conventional methods, such as linear (or nonlinear) discrimination techniques or statistical methods.

2. The method automatically detects and reduces redundant variables.

3. Classification rules in the form of VL_1 expressions are very easy to interpret by humans and at the same time very convenient to evaluate by computers (especially using parallel or parallel-sequential techniques).

4. Although the synthesis of minimal VL_1 expressions is a complex combinatorial problem, experimental results from program AQVAL/1 prove that the execution time and memory requirements in solving 'average size' problems are quite acceptable.

5. It is also worth noting that the method can be extended to cover the case of not completely specified events, i.e., when some measurements are missing (what is a common situation in applications), and, also, that the classification rules (obtained based on some given data) can be easily modified if new information is given which is contradictive to the rule.

References

1. Michalski, R.S., "A Variable-Valued Logic System as Applied to Picture Description and Recognition," GRAPHIC LANGUAGES, Proceedings of the IFIP Working Conference on Graphic Languages, ed. F. Nake, A. Rosenfeld, Vancouver, Canada, May 1972.

2. Sebestyen, G.S., DECISION-MAKING PROCESSES IN PATTERN RECOGNITION, The Macmillan Co., New York, 1962.

3. Patrick, E.A., FUNDAMENTALS OF PATTERN RECOGNITION, Prentice-Hall, Inc., New Jersey, 1972.

4. Michalski, R.S., "On the Quasi-Minimal Solution of the General Covering Problem," Proceedings of the V International Symposium on Information Processing (FCIP 69), Vol. A3 (Switching Circuits), Yugoslavia, Bled, October 8-11, 1969.

5. Michalski, R.S. and McCormick, B. H., "Interval Generalization of Switching Theory," Proceedings of the Third Annual Houston Conference on Computer and System Science, Houston, Texas, April 26-27, 1971 (an extended version: Department of Computer Science Report No. 442, University of Illinois at Urbana-Champaign, May 3, 1971)

6. Michalski, R.S., "AQVAL/1--Computer Implementation of a Variable-Valued Logic System VL, and Examples of its Application to Pattern Recognition," submitted for presentation at the International Joint Conference on Pattern Recognition, October 30-November 1, 1973, Washington, D. C.

7. Michalski, R.S., "A Geometrical Model for the Synthesis of Interval Covers," Department of Computer Science Report No. 461, University of Illinois at Urbana-Champaign, June 24, 1971.

8. Bongard, M.M., PROBLIEMA UZNAVANIA, Moskva, 1967 (English translation: PATTERN RECOGNITION, Spartan Books, 1970).

9. Gagliardo, E., "Abicsofos 1972," Technical Report No. 47, Department of Mathematics, Oregon State University, Corvallis, Oregon, April 1973.

10. Necher, N., MANY-VALUED LOGIC, McGraw-Hill, New York, 1969.

11. Nutter, R.S., "Function Simplification Techniques for Postian Multi-Valued Logic Systems," Dissertation, West Virginia University, 1971.

SET	NO. OF THE OBJECT	REPRESENTATION (EVENTS) x_6	x_7	x_8	
$r(\emptyset_1)$	1	1	2	1	e^{41}
		1	3	3	e^{47}
		2	3	3	
		3	5	3	
		3	6	3	
		5	6	1	e^{185}
	2	5	0	3	e^{163}
		0	6	3	
	3	1	0	3	e^{35}
		0	5	3	
		0	6	3	
		5	6	1	e^{185}
	4	1	7	3	e^{63}
		7	5	3	
		7	6	3	
		5	6	1	e^{185}
$r(\emptyset_0)$	1	1	3	3	e^{47}
		3	5	2	
		3	6	2	
		5	6	1	e^{185}
	2	4	0	3	e^{163}
		0	7	3	
	3	1	0	3	e^{35}
		5	0	1	
		0	6	2	
	4	1	2	1	e^{41}
		1	7	3	e^{63}
		2	7	3	
		7	5	2	
		7	5	2	
		5	6	1	e^{185}

Table 1

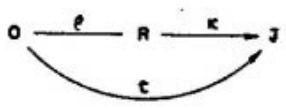


FIGURE 1

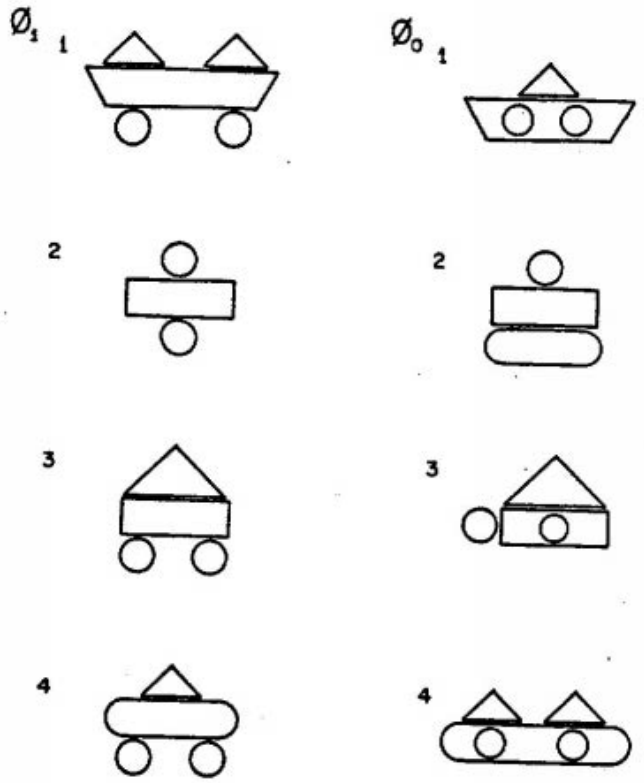
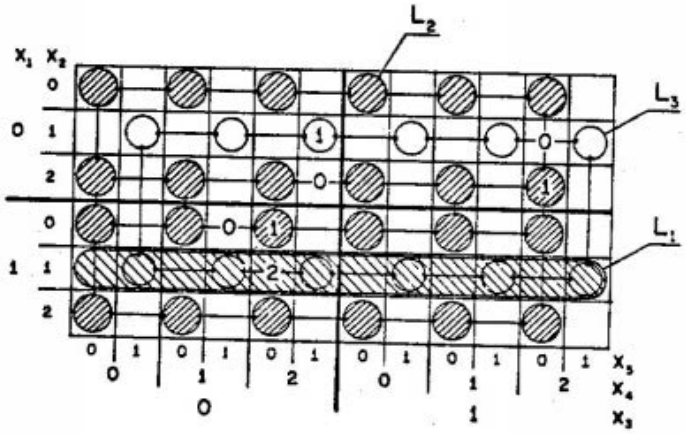


FIGURE 2



L_1 =: 2[x_1=1][x_2=1]

L_2 =: 1[x_2=0,2][x_5=0]

L_3 =: 1[x_2=1][x_5=1]

(L_j =: T means that set of cells L_j corresponds to term T)

FIGURE 3

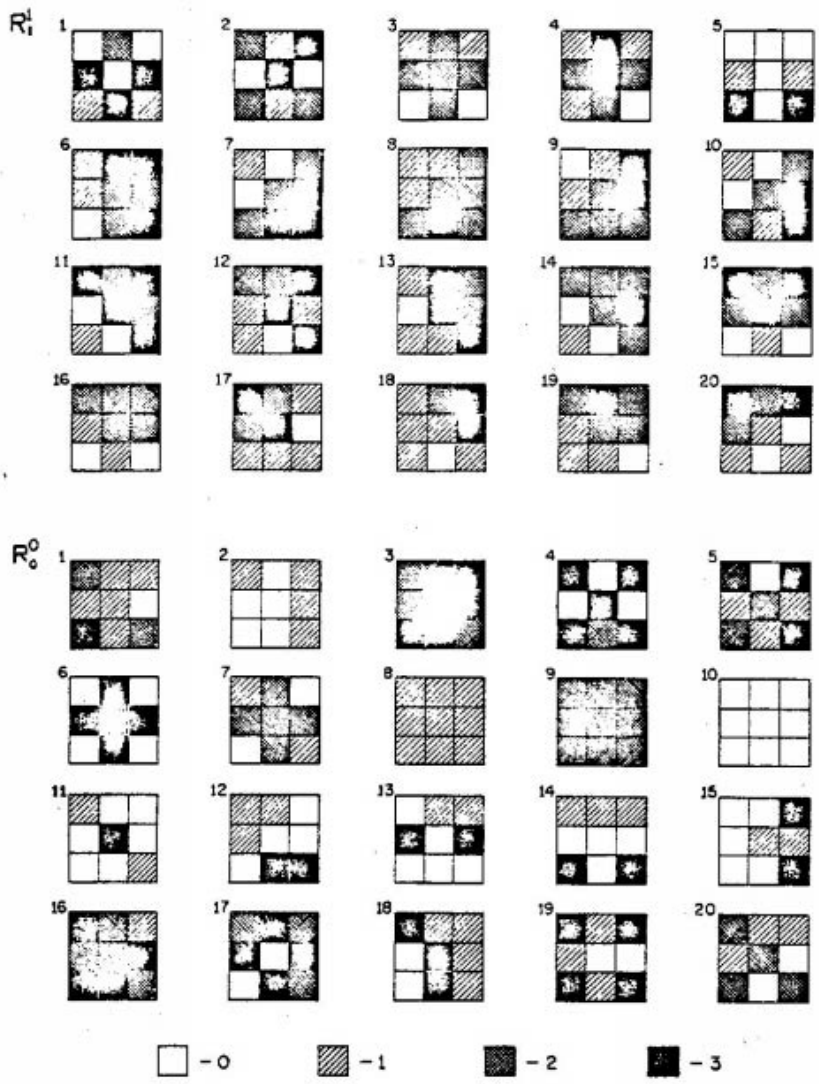


FIGURE 4

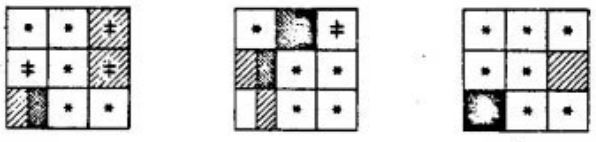


FIGURE 5