

A VARIABLE-VALUED LOGIC SYSTEM  
AS APPLIED TO PICTURE  
DESCRIPTION AND RECOGNITION

by

*Ryszard S. Michalski*

Chapter in the book, *Graphic Languages*, F. Nake and A. Rosenfeld,  
(Editors), North-Holland Publishing Company, pp. 20-47, 1972.

# *GRAPHIC LANGUAGES*

---

Proceedings of the  
IFIP Working Conference on  
Graphic Languages

edited by

**F. NAKE**

*University of British Columbia Vancouver, B.C., Canada*

**A. ROSENFELD**

*University of Maryland College Park, Md., U.S.A.*



1972

NORTH-HOLLAND PUBLISHING COMPANY — AMSTERDAM • LONDON

Reprinted from 'GRAPHIC LANGUAGES' - Editors: F. NAKE and A. ROSENFELD

A VARIABLE-VALUED LOGIC SYSTEM  
AS APPLIED TO PICTURE DESCRIPTION  
AND RECOGNITION

Ryszard S. Michalski  
Department of Computer Science  
University of Illinois  
Urbana, Illinois, U.S.A.

**Abstract:** The paper introduces a concept of a variable-valued logic system, defines a particular system VL, and demonstrates, by examples, its application as a language for describing complex graphical objects and also as a tool for making inferences about significant properties of the objects or classes of objects. (Included examples show how to use the system to describe a picture treated as a structure of interrelated components, to deduce a simple rule characterizing one class of patterns as opposed to another and to synthesize a minimal set of filters for the discrimination of a collection of textures.) A brief summary of the computer implementations of the developed concepts is also included.

## 1. INTRODUCTION

In the conventional approach to designing graphical languages, the problems of obtaining the simplest of possible equivalent descriptions of an object (according to certain defined criteria) have not received much consideration. Yet, in the application of the descriptions to the recognition of objects, the simplicity of descriptions plays an important role. Another problem related to the recognition of objects, in particular, of pictures, is to provide a formal tool for selecting the most 'significant' features of the described objects.

A possibility of having a formal system with mechanisms for obtaining the simplest (in a clearly defined sense) descriptions of objects, as well as for finding their most characteristic features, has a practical importance. The purpose of the paper is to investigate this possibility. We present here a formal system with a few simple operations (or a language in a broader sense) which--when supplied with primitives and their relations (found in a described object or class of objects)--produces a description which is minimal in a well-defined sense. The system can also be used to deduce the simplest rule for recognizing one class in the context of an other class (or classes) of objects. This system belongs to the domain of logic rather than to that of formal languages and is called variable-valued logic. Actually, in the paper we describe one particular system of the variable-valued logic, namely VL<sub>1</sub>. It is an outgrowth of my work with Professor Bruce McCormick on the so-called "interval and cartesian covers". A number of concepts included in the system are directly related to the concepts in our common reports (Michalski and McCormick (1971c, 1972b)).

Since the description of VL<sub>1</sub> is not available elsewhere, it was necessary to include it here, and therefore Chapter 2 is devoted to the formal definition of the system. Chapter 3 describes briefly the problem of synthesizing formulas in the system (which satisfy certain cost functionals) and summarizes the computer implementations of the developed concepts. Chapter 4 gives a number of simple examples which show, in particular, the application of the system to inferring, based on logical principles, a simple rule which characterizes one class of objects as compared to another or to finding a minimal set of filters for discrimination of textures. The last example is used to demonstrate how to apply VL<sub>1</sub> to describing a graphical object, viewed as a structure of interrelated components. (We adopted this order of examples since, e.g., in this last example we assume a texture as a given attribute of a component of the described object; and this

assumption is more acceptable in the context of the previous example which considered a problem of the recognition of textures.)

Before the formal exposition of the system, we feel that a few basic assumptions underlying its definition should be explained.

First, we assumed that the syntax and semantics of the system should be inseparable parts of the whole, and therefore the formation and interpretation rules of the system are included as parts of its definition. Second, we did not assume any concrete primitive elements in the system, but just numbers which can be interpreted as 'numerical names' of anything we want. We consider the system as a mechanism which receives these numerical names as its input and then manipulates them in a certain way in order to logically infer some more general properties of the input data. In order to develop such an automatic inference system, we sought help from the developments of formal logic. Results from many-valued logic seemed to be especially interesting since the attributes (or tests) with which we describe objects often have several values (or outcomes).

The direct application of many-valued logic to our problems meets, however, different kinds of difficulties. One of them is, for example, that in, say,  $k$ -valued logic, when applied to describing objects or situations, every proposition is assumed to be  $k$ -valued--with no regard to the fact that some of the propositions can intrinsically require different numbers of truth-values. (E.g., suppose we wanted to apply  $k$ -valued logic formalism to describing a human being. One of the attributes we may use, hair color, is many-, say,  $k$ -valued. Another, sex, is intrinsically two-valued. Thus difficulties arise with the interpretation of the  $k-2$  unused values of the sex attribute, and also other unnecessary problems can occur.)

We found it useful to assume that every proposition (or attributes of objects involved in a proposition) should be allowed to have its own independent number of truth-values. This number is chosen according to the practical interpretation or particular purpose. Moreover, to make computation feasible, it is assumed that this number is chosen not only finite but as small as is possible without losing considerably in practical validity. A reason for this is that an automatic inference system, in order to deduce relevant features or variables for a problem-oriented description, should have access to a large number of variables. A possibility of handling many variables can, however, be provided only if they are crudely represented.

(As an illustrative example, consider a doctor who is making a diagnosis based on observed symptoms which include, e.g., the temperature of a patient's body. We suspect, that although the range of variability of temperature is, theoretically, continuous (between, say,  $94$  and  $108^\circ$  F.), a doctor thinks in categories like--normal, below normal, high, very high, etc., thus neglecting the exact measurements he is given.)

Another assumption is, that in a logic system which we want to apply to problems such as pattern recognition, the primitive functions should be selected based on the practical considerations, and--as a general rule--their number should be as large as is computationally feasible (the opposite of the usual tendency). This will allow the system to select those primitive functions (from a set of available functions) which allow it to describe a given object in the simplest way. An analogy to this is an observation, e.g., that a person, in recognizing objects, usually applies very simple decision rules but uses different 'primitive functions' each time, rather than complicated rules with the same fixed 'primitive functions'.

To conclude, it may be worth noting that the VL system (or rather developed programs based on it) can also be applied to a number of problems in artificial intelligence, switching theory, graph theory, information retrieval, data reduction, map coloring, constructing decision graphs, etc.

## 2. BASIC DEFINITIONS

2.1 Definition of a Variable-Valued Logic System

A variable-valued logic system (a VL system) is an ordered quintuple:

$$(X, H, S, R_F, R_I) \quad (1)$$

where:

- $X$  is a finite non-empty (f.n.) set of input variables, whose domains are any f.n. sets, called input name sets,  
 $H$  is a f.n. set, called output name set,  
 $S$  is a f.n. set of improper symbols,  
 $R_F$  is a f.n. set of formation rules which define well formed formulas (wffs) in the VL System (or VL formulas). A string of elements from  $X$ ,  $H$  or  $S$  is a wff if and only if it can be derived from a finite number of applications of the formation rules.  
 $R_I$  is a f.n. set of interpretation rules which give an interpretation to the VL formulas.

In the paper we will focus our attention on one specific system, called  $VL_1$ .

2.2 Definition of the  $VL_1$  System

$VL_1$  is a variable-valued logic system  $(X, H, S, R_F, R_I)$ , where:

- $X$  is a set of variables  $x_1, x_2, x_3, \dots, x_n$  whose domains are sets  $H_1, H_2, H_3, \dots, H_n$ , respectively, where  
 $H_i = \{0, 1, \dots, M_i\}$ ,  $i = 1, 2, \dots, n$ ,  
 $M_i$  - a natural number. (2)

- $H$  is a set of constants  
 $H = \{0, 1, \dots, M\}$ ,  $M$  - a natural number. (3)

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

$S$  consists of 11 improper symbols:

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

$R_F$  is the set of the formation rules:

1. A primitive constant from  $H$  standing alone is a wff.
2. A form  $[x_i \# c]$ , where  $i \in \{1, 2, \dots, n\}$ ,  $\# \in \{=, \neq\}$ ,  $c$  -- a sequence of elements of  $H_i$  separated by ',' or ':' is a wff. If the separation is by ':', then it is assumed that the first constant is smaller than the second, e.g. if  $c$  is  $c_1 : c_2$ , then  $c_1 < c_2$ .
3. If  $V, V_1$  and  $V_2$  are wffs then  $\bar{V}, (V), V_1 \wedge V_2$  (written also as  $(V_1 V_2)$ ),  $V_1 \vee V_2$  and  $[V_1 \# V_2]$ ,  $\# \in \{=, \neq\}$  are also wffs.

A wff in  $VL_1$  is also called a  $VL_1$  formula. Forms  $[x_i \# c]$  and  $[V_1 \# V_2]$  are called selectors (the former form is also called a simple selector).

In the simple selector  $[x_i \# c]$ ,  $c$  is called the range of  $x_i$ .

If '#' is '=' then the range is called inclusive, otherwise exclusive.

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

1. The value  $v(c)$  of a constant  $c, c \in H$ , is  $c$ , which is denoted by  $v(c) = c$ .
2.  $v([x_i \# c]) = \begin{cases} \# & \text{if } x_i \# c \\ 0 & \text{otherwise} \end{cases}$

where  $x_i \# c$  means  $x_i = c$  or  $x_i \neq c$  and the relation  $x_i = c$  ( $x_i \neq c$ ) is satisfied if the value of the variable  $x_i$  is (is not) one of the elements in the sequence  $c$ . If two elements in  $c$  are separated by ':' then the relation is also satisfied if the value of  $x_i$  is (is not) between the above two elements. E.g., if  $c$  is a sequence 1,3:6,8 then

$$v([x_i = 1,3:6,8]) = \begin{cases} \# & \text{if } x_i \text{ equals } 1, 3, 4, 5, 6, \text{ or } 8 \\ 0 & \text{otherwise} \end{cases}$$

and

$$v([x_i \neq 1,3:6,8]) = \begin{cases} \# & \text{if } x_i \text{ equals } 0, 2, 7 \text{ (assuming } \# = 8) \\ 0 & \text{otherwise} \end{cases}$$

The selector  $[x_i \# c]$  is said to be satisfied if  $x_i \# c$ .

3.  $v([V_1 \# V_2]) = \begin{cases} \# & \text{if } v(V_1) \# v(V_2) \\ 0 & \text{otherwise} \end{cases}$   
where the meaning of  $\#$  and the definition of a selector being satisfied is the same as in 2.
4.  $v(\bar{V}) = \# - v(V)$
5.  $v(V_1 V_2) = \min(v(V_1), v(V_2))$   
 $V_1 V_2$  is called a product (or conjunction) of  $V_1$  and  $V_2$ .
6.  $v(V_1 \vee V_2) = \max(v(V_1), v(V_2))$   
 $V_1 \vee V_2$  is called a sum (or disjunction) of  $V_1$  and  $V_2$ .
7. Parentheses ( ) have their usual meaning, i.e. they denote a part of a formula which is to be evaluated as a whole.

Thus, the above interpretation rules interpret  $VL_1$  formulas as expressions of functions:

$$f: H_1 \times H_2 \times \dots \times H_n \longrightarrow H \quad (4)$$

The set  $H_1 \times H_2 \times \dots \times H_n, H_i = \{0, 1, \dots, \#_i\}, i = 1, 2, \dots, n$  will be denoted by  $E(h_1, h_2, \dots, h_n)$ , where  $h_i = c(H_i)^* = \#_i + 1$ , or briefly by  $E$ , and called the universe of events. The elements of  $E$ , vectors  $(x_1, x_2, \dots, x_n), x_i \in H_i, i = 1, 2, \dots, n$ , will be called events and denoted by  $e^j, j = 0, 1, 2, \dots$ . The universe  $E$  has  $h = h_1 h_2 \dots h_n$  events, thus:

$$E(h_1, h_2, \dots, h_n) = \left\{ e^j \right\}_{j=0}^{\#} \quad (5)$$

where  $\# = h-1$  ( $x_i \in H_i$  denotes, for simplicity, that values of  $x_i$  are from  $H$ ).

We will assume that the values of  $j$  are given by a function

$$\gamma: E \rightarrow \{0, 1, \dots, \#\} \quad (6)$$

such that

$$j = \gamma(e) = x_n + \sum_{k=n-1}^1 x_k \prod_{i=n}^{k+1} h_i \quad (7)$$

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

$\gamma(e)$  is called the number of the event  $e$ . E.g., an event  $e = (2,1,1)$  in the universe  $E(4,3,2)$  has the number  $\gamma(e) = 1 + 1 \cdot 2 + 2 \cdot 3 \cdot 2 = 15$ . It is easy to see that, given  $\gamma(e)$ , one can determine  $e$ .

Examples of  $VL_1$  Formulas:

$$3[x_1 = 2,4:6][x_4 = 2] \vee 2[x_2 = 1:3][x_4 \neq 0] \vee 1[x_3 = 3] \quad (8)$$

$$3[2[x_1 = 2][x_3 \neq 2,3] = 2[x_2 \neq 3]] \vee 2[x_2 = 2:5](1 \vee 3[x_1 = 0]) \quad (9)$$

$$(3 \vee 2[x_2 \neq 2])(3[x_1 = 2] \vee 1[x_2 \neq 0]) \quad (10)$$

The formula (8) can be interpreted as an expression of a function

$$f: E(7,6,4,3) \longrightarrow \{0,1,2,3\} \quad (11)$$

assuming that  $c(H_i) = 7,6,4,3$  for  $i = 1,2,3,4$ , and  $c(H) = 4$ . The formula assumes the value 3 if the selectors  $[x_1 = 2,4:6]$  and  $[x_4 = 2]$  are satisfied (independent of the values of  $x_2$  and  $x_3$ ). The formula assumes the value 2 if the previous condition does not hold (since  $V_1 \vee V_2$  means  $\max(V_1, V_2)$ ) and the selectors  $[x_2 = 1:3]$  and  $[x_4 \neq 0]$  are satisfied. The formula assumes the value 1 if both of the previous conditions do not hold and the selector  $[x_3 = 3]$  is satisfied. If none of the above conditions hold, the formula assumes the value 0.

Note that selectors correspond to certain sets of events (event sets), namely those which satisfy them (thus, selectors select event sets). A  $VL_1$  formula is called an interval  $VL_1$  formula if the range  $e$  in each of its selectors is in the form  $c_1:c_2$  or  $c$  (this formula has direct correspondence to interval covers introduced by Michalski and McCormick (1971c)). A product of selectors and constants is called a term. A  $VL_1$  formula which is a sum (disjunction) of terms (e.g.(8)) is called a disjunctive  $VL_1$  formula (DVL<sub>1</sub>). In the sequel we will mainly consider DVL<sub>1</sub> formulas, thus  $VL_1$  will mean DVL, if it is not stated otherwise.

In practice we usually deal with functions which may have an unspecified value for some events, that is with functions.

$$f: E \longrightarrow H \cup \{*\} \quad (12)$$

where  $*$  denotes an unspecified value ('don't care').

An incompletely specified function  $f$  can be considered equivalent to a set  $\{f_i\}$  of completely specified functions  $f_i$ , each determined by a certain assignment of specified values (i.e. values from  $H$ ) to events  $e$  for which  $f(e) = *$ . A  $VL_1$  formula which expresses any of these functions  $f_i$  will be accepted as an expression for  $f$ .

Functions of the type (12) will be called variable-valued logic functions (VL functions).

It is easy to see that special cases of  $VL_1$  are e.g. a two-valued Boolean algebra (if  $h_1 = h_2 = \dots = h_n = 2$ ;  $[x_1 = 1]$  and  $[x_1 = 0]$  would correspond to  $x_1$  and  $\bar{x}_1$ , respectively), Multivalued Disjoint Post Systems (Nutter (1971)), a Multi-Valued Logic System described by Givone and Snelsire (1968).

2.3 A Geometrical Representation of VL Functions

A VL function can be geometrically represented using a generalized logical diagram (GLD) described by Michalski (1971a). For example, Fig. 1 represents a function:

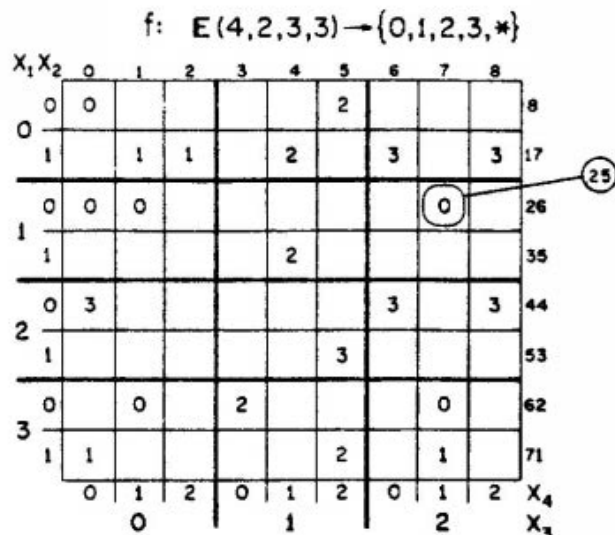


Figure 1

Cells of the diagram correspond to events in  $E(4,2,3,3)$ . This correspondence is self-explanatory, e.g. cell 25 corresponds to event  $e^{25} = (1,0,2,1)$ . Numbers on the top and the right of the diagram are the numbers of the events to which the cells correspond (note a lexicographical order of the numbers). Cells are marked by values of the function  $f$  applied to the events to which the cells correspond. Empty cells denote events for which  $f$  equals  $*$ .

2.4 Definition of Minimal VL<sub>1</sub> Formula

A given function  $f$  can be expressed by a number of different VL<sub>1</sub> formulas (even for small  $n$ ,  $h_1$  and  $h$  this number can be very large). For example, the function  $f$  in Fig. 1 is represented by a VL<sub>1</sub> formula:

$$\begin{aligned}
 f: & \quad 3[x_1 = 2][x_2 = 0][x_4 = 0,2] \vee 3[x_1 = 2][x_2 = 1][x_3 = 1] \vee \\
 & \quad 3[x_1 = 0,3][x_3 = 2][x_4 = 0,2] \vee 2[x_1 = 0,1][x_2 = 0][x_3 = 1,2][x_4 = 0,2] \\
 & \quad \vee 2[x_3 = 1][x_4 = 0,1] \vee 2[x_1 = 3][x_2 = 1][x_3 = 1][x_4 = 1,2] \vee \\
 & \quad 1[x_1 = 0][x_2 = 1][x_3 = 0][x_4 = 1,2] \vee 1[x_1 = 1,3][x_2 = 1][x_3 = 0,1] \\
 & \quad \vee 1[x_2 = 1][x_3 = 1,2][x_4 = 0,1] \tag{13}
 \end{aligned}$$

( $f$ :  $V$ , where  $V$  is a VL formula, means that  $V$  is an expression for  $f$ ).

The same function  $f$  is, however, also represented by:

$$f: \quad 3[x_3 = 2][x_4 \neq 1] \vee 3[x_1 = 2] \vee 2[x_3 = 1] \vee 1[x_2 = 1] \tag{14}$$



Fig. 2 shows the event sets which correspond to terms of (14). ( $L = S$ , where  $S$  is a selector or a product of selectors, means that  $L$  is an event set which satisfies  $S$ ).

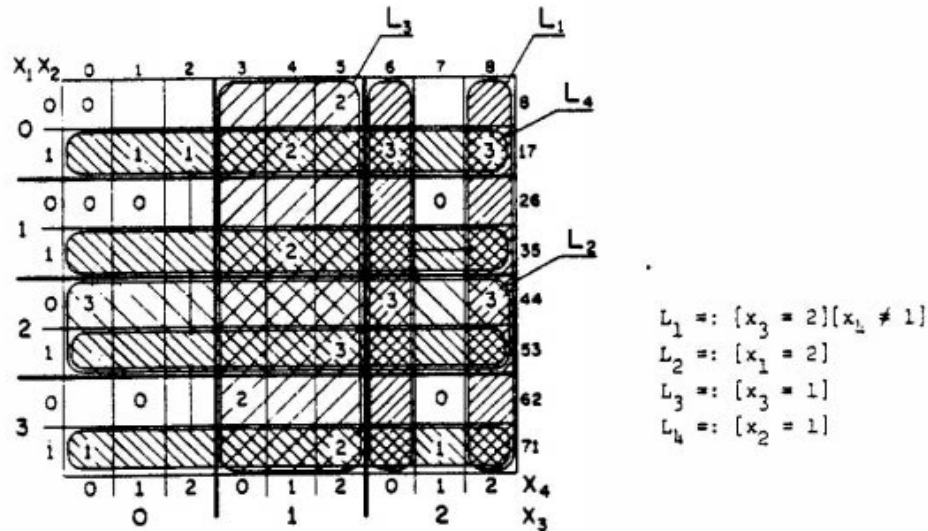


Figure 2

The problem arises of how to find a  $VL_1$  formula for a given  $f$ , which minimizes an assumed cost functional. This functional can involve different characteristics of  $VL_1$  formulas (e.g. the number of terms, the number of selectors, the sum of ranges of variables in the selectors, the different costs of selectors with different variables, etc.) depending on the applications.

From a practical as well as a computational point of view, it is convenient to assume a functional  $A(V)$  to be in the form:

$$A(V) = \langle a_1(V), a_2(V), \dots, a_k(V) \rangle \quad (15)$$

where  $a_i(V)$ ,  $i = 1, 2, \dots, k$  are different single-valued characteristics (attributes) of a formula  $V$  expressing  $f$ . A formula  $V$  is said to be a minimal formula

under the functional  $A$  for the function  $f$  iff  $A(V) \leq A(V_j)$ ,  $j = 1, 2, 3, \dots$ ,

where  $\{V_j\}$  are all possible formulas expressing  $f$

and  $\leq$  denotes a lexicographical order, i.e.

$$A(V) \leq A(V_j) \text{ if } \begin{cases} a_1(V) < a_1(V_j) \\ \text{or } a_1(V) = a_1(V_j) \text{ and } a_2(V) < a_2(V_j) \\ \text{or } \dots \text{ and } a_k(V) < a_k(V_j) \end{cases}$$

Here we will primarily consider a functional  $\langle t(V), s(V) \rangle$  or briefly the  $\langle t, s \rangle$ , where  $t = t(V)$ ,  $s = s(V)$  are the number of terms and the number of selectors in  $V$ , respectively.

Thus, a minimal  $VL_1$  formula for a given  $f$  is a  $DVL_1$  formula which has the

minimum number of terms and the minimum number of selectors for that number of terms.

The problem of the synthesis of minimal  $VL_1$  formulas is discussed in Section 3.

2.5 Complex  $VL_1$  Formulas

As was previously mentioned, a  $VL_1$  formula expresses a function

$$f: E \longrightarrow H \quad (16)$$

In practical applications we may be interested in expressing not just one function  $f$  but a family of functions with the same domain  $E$ :

$$f: E \longrightarrow H \quad (17)$$

where

$$\begin{aligned} f &= ({}^1f, {}^2f, \dots, {}^mf), \quad {}^kf: E \longrightarrow {}^kH, \quad k = 1, 2, \dots, m, \\ {}^kH &= \{0, 1, 2, \dots, {}^kH\} \\ H &= {}^1H \times {}^2H \times \dots \times {}^mH \end{aligned}$$

Thus:

$$f: H_1 \times H_2 \times \dots \times H_m \longrightarrow {}^1H \times {}^2H \times \dots \times {}^mH \quad (19)$$

A function  $f$  is called a complex (or multiple-output) VL function. It can be expressed by a single  $VL_1$  expression by extending the set  $X$  with an additional variable  $y$  whose domain is  $\{1, 2, \dots, m\}$  and assuming that

$$H = \max({}^1H, {}^2H, \dots, {}^mH).$$

The role of the variable  $y$  is expressed by an additional interpretation rule:

8.  $v(V[y = c])$  is interpreted as follows:

the value  $v(V)$  is given to functions  $\{{}^yf \mid y = c\}$ .

For example, if  $V_1, V_2$  and  $V_3$  are  $VL_1$  formulas which do not include the variable  $y$ , then

$$V_1[y = 1,2,3] \vee V_2[y = 3,4] \vee V_3[y = 1,3] \quad (20)$$

is interpreted as an expression of a function  $f = ({}^1f, {}^2f, {}^3f, {}^4f)$  where

$${}^1f: V_1 \vee V_3, \quad {}^2f: V_1, \quad {}^3f: V_1 \vee V_2 \vee V_3, \quad {}^4f: V_2 \quad (21)$$

The selector  $[y = c]$  is called a function selector. A  $VL_1$  formula which includes function selectors is called a complex  $VL_1$  formula.

We extend a function  $f$  (18) to an incompletely specified function by assuming that:

$$f: E \longrightarrow ({}^1H \cup \{*\}) \times ({}^2H \cup \{*\}) \times \dots \times ({}^mH \cup \{*\}) \quad (22)$$

3. SYNTHESIS OF  $VL_1$  FORMULAS

3.1 Basic Concepts from Covering Theory

The synthesis of a  $VL_1$  formula for expressing a given function  $f$  is based on the results of covering theory (Michalski (1969ab, 1971ac, 1972b)). This theory deals with the problems of expressing sets as unions of certain standard subsets

('building blocks') called complexes. One of the basic concepts is that of a cover of a set against another set.

Let  $E^1$  and  $E^0$  be disjoint event sets in a universe  $E$  and sets

$$L = \bigcap_{i \in I} X_i^{a_i} \quad (23)$$

where  $A_i \in H_i, I \in \{1, 2, \dots, n\}$  (24)

$$X_i^{a_i} = \{e = (x_1, x_2, \dots, x_i, \dots, x_n) \mid x_i \in A_i\} \quad (25)$$

'building blocks' which are used to express event sets. Event sets  $L$  (23) are called cartesian complexes and sets  $X_i^{a_i}$  (25) -- cartesian literals.

A cartesian cover  $C(E^1 | E^0)$  of set  $E^1$  against  $E^0$  is a set  $\{L_i\}$  of cartesian complexes  $L_i$  such that

$$E^1 \subseteq \bigcup L_i \subseteq E \setminus E^0 \quad (26)$$

If sets  $A_i$  are restricted to sequences of consecutive numbers from  $H_i$ , then the complexes  $L$  are called interval complexes and denoted as

$$\bigcap_{i \in I} {}^{a_i, b_i} X_i \quad (27)$$

where  ${}^{a_i, b_i} X_i = \{e = (x_1, x_2, \dots, x_i, \dots, x_n) \mid a_i \leq x_i \leq b_i\}$

A cover which consists of only interval complexes is called an interval cover.

It is easy to see that a cartesian cover  $C(E^1 | E^0)$  (or an interval cover  $I(E^1 | E^0)$ ) is equivalent to a  $VL_1$  formula (or an interval VL formula) which expresses a function  $f: E \rightarrow \{0, 1, *\}$ , assuming that  $E^1 = \{e \mid f(e) = 1\}$  and  $E^0 = \{e \mid f(e) = 0\}$ . For example, if  $C(E^1 | E^0) = X_1^{1,3} \cup X_2^{0,2,3} \cup X_3^{1,2,4}$  and  $I(E^1 | E^0) = {}^{0,2} X_1 \cup {}^{1,3} X_2 \cup {}^{1,2,4} X_3$

then the corresponding  $VL_1$  formulas are:

$$V_C = [x_1 = 1,3][x_3 = 0,2,3] \vee [x_2 = 1,2,4] \text{ and } V_I = [x_1 = 0:2] \vee [x_2 = 1:3][x_4 = 0]$$

### 3.2 A Brief Description of a Synthesis Algorithm

A VL function  $f$  (12) can be specified by a family of event sets:

$$\{F^M, F^{M-1}, \dots, F^0\} \quad (28)$$

where:  $F^k = \{e \mid f(e) = k\}, k = 0, 1, \dots, M.$

Let  $M(E_1 | E_2)$  be a minimal cartesian cover of  $E_1$  against  $E_2$  which is defined as a cartesian cover with the minimum number of complexes and minimum number of literals for that number of complexes. Since cartesian complexes correspond to terms and literals to selectors in a corresponding  $VL_1$  formula, the cover  $M(E_1 | E_2)$  corresponds to a minimal  $VL_1$  formula under the functional  $\langle t, s \rangle$ .

An algorithm for constructing a minimal  $VL_1$  formula under  $\langle t, s \rangle$  for a given function  $f$  is:

1. Determine the minimal cartesian covers:

$$\begin{aligned} M^k &= M(F^k | F^{k-1} \quad F^{k-2} \quad \dots \quad F^0) \\ M^{k-1} &= M(F^{k-1} | F^{k-2} \quad F^{k-3} \quad \dots \quad F^0) \\ &\vdots \\ &\vdots \\ &\vdots \\ M^1 &= M(F^1 | F^0) \end{aligned}$$

2. Determine the products of selectors which are equivalent to complexes in the covers  $M^k, M^{k-1}, \dots, M^1$
3. Multiply the products equivalent to complexes in  $M^k$  --by  $k$ , in  $M^{k-1}$  --by  $k-1, \dots$ , in  $M^1$  --by 1.
4. The union of all terms thus obtained is a minimal  $VL_1$  formula under  $\langle t, s \rangle$  for the function  $f$ .

For a proof see Michalski (1972a). In the case of the synthesis of a minimal interval  $VL_1$  formula, the procedure would be the same, except that interval covers rather than cartesian covers would be constructed.

### 3.3 AQ Programs

The synthesis of a minimal  $VL_1$  formula was, in 3.2, reduced to the iterative synthesis of minimal cartesian covers. The synthesis of minimal covers is, however, computationally feasible only in rather simple cases. In general, such synthesis can require the enumeration of an extensive set of possibilities (the proof of this is similar to the proof by Zhuravlev (1960) on the necessity of enumeration in the minimization of switching functions). But this set of possibilities (after applying all possible reduction techniques) is usually too large to make the enumeration feasible. Therefore, the only realistic approach here is to seek approximate solutions (except for simple, academic problems).

A good method for an approximate solution should drastically reduce the number of operations required for a minimal solution and produce a cover which has a high probability of being close to the minimal one. Moreover, it is highly desirable to also have an estimate of the distance between the obtained solution and the minimal one. The only existing algorithm (to the best of the author's knowledge) which satisfies all 3 criteria is an algorithm  $A^q$  based on the principle of disjoint stars (Michalski 1969a, 1969b).

This algorithm has been quite successful in the minimization of switching functions of many variables (Michalski, Kulpa (1971)) as well as in the synthesis of interval and cartesian covers (Michalski, Tareski (1972d)). The algorithm (valid for any kind of complexes) produces a so-called quasi-minimal cover, which is an approximately minimal or minimal one, and gives an estimate  $\Delta$  of the maximal number of complexes in which the obtained and minimal covers can differ:

$$c(M^q) - c(M) \leq \Delta \quad (29)$$

where  $M^q$  is a quasiminimal cover

$M$  is a minimal cover.

Based on this algorithm the following programs\* have been developed at the University of Illinois.

1. AQ--3i For the synthesis of interval covers.
2. AQ--4c For the synthesis of cartesian and interval covers (or mixed, where some literals can be cartesian and some interval).
3. AQ--5cm For the synthesis of cartesian covers of a disjoint family of event sets (covers of each set against all the others).
4. AQVAL/1 For the synthesis of quasi-minimal  $VL_1$  expressions under a functional  $\langle a_1, a_2, \dots \rangle$  where  $a_i, i \leq 7$ , can be chosen (in any order), depending on the applications, from a set of 7 attributes, such as:
  1.  $t(V)$ --the number of terms in  $V$ ,
  2.  $s(V)$ --the number of selectors in  $V$ ,
  3.  $g(V)$ -- the 'degree of generalization' defined as:

$$g(V) = \frac{1}{g_0} \sum_{k=1}^M \sum_{l=1}^{s_k} g(L_{kl}) \quad (30)$$

where  $g(L_{kl}) = \frac{c(L_{kl})}{c(L_{kl} \cap F^k)} - 1$

$L_{kl}$ --a complex corresponding to the  $l$ th term of the terms in  $V$  with the constant  $k$ ,

$s_k$ --number of terms with the constant  $k$ ,

$g_0$ --total number of terms in  $V$

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

4.  $r(V)$ --a 'range coefficient' defined as:

$$r(V) = \sum_{j=1}^{s(V)} r_j, \text{ where } r_j \text{ is the number of elements in the range of the } j\text{th selector,}$$

and some others (Michalski (1972a)).

5. AQVAL/2

For the synthesis of a quasi-minimal family of  $VL_1$  expressions (a complex  $VL_1$  expression) under the functional used in AQVAL/1.

---

\*These programs do not include programs based on the same algorithm for the synthesis of switching circuits developed at the Polish Academy of Sciences.

All of the above programs have been written in PL/1 for the IBM 360/75. (Their description is given in a number of reports: AQ--3i by Michalski and Tareski (1972d), AQ--4c and AQ--5cm by Michalski and Raulefs (1972c), AQVAL/1 and AQVAL/2 by Michalski (1972a)). To give an estimate of their size, e.g. AQ--3i has ~ 660 PL/1 statements and AQVAL/2 ~ 1400 PL/1 statements.

4. EXAMPLES OF APPLICATIONS OF VL<sub>1</sub> TO PATTERN RECOGNITION

In this chapter we will give some examples of the application of the VL<sub>1</sub> system to pattern recognition, in particular to the synthesis of efficient classifying rules, to the selection of features (based on a logical deduction as opposed to e.g. a statistical approach), to texture discrimination and to the description of complex graphical objects. These examples are very simple--not encyclopedic in the range of possible applications but sufficient to provide an insight into the principles and a general appreciation of the techniques used.

Example 1

Fig. 3 presents two sets of patterns: F<sup>1</sup> and F<sup>0</sup>. Find a simple rule for distinguishing patterns of the first set from patterns of the second.

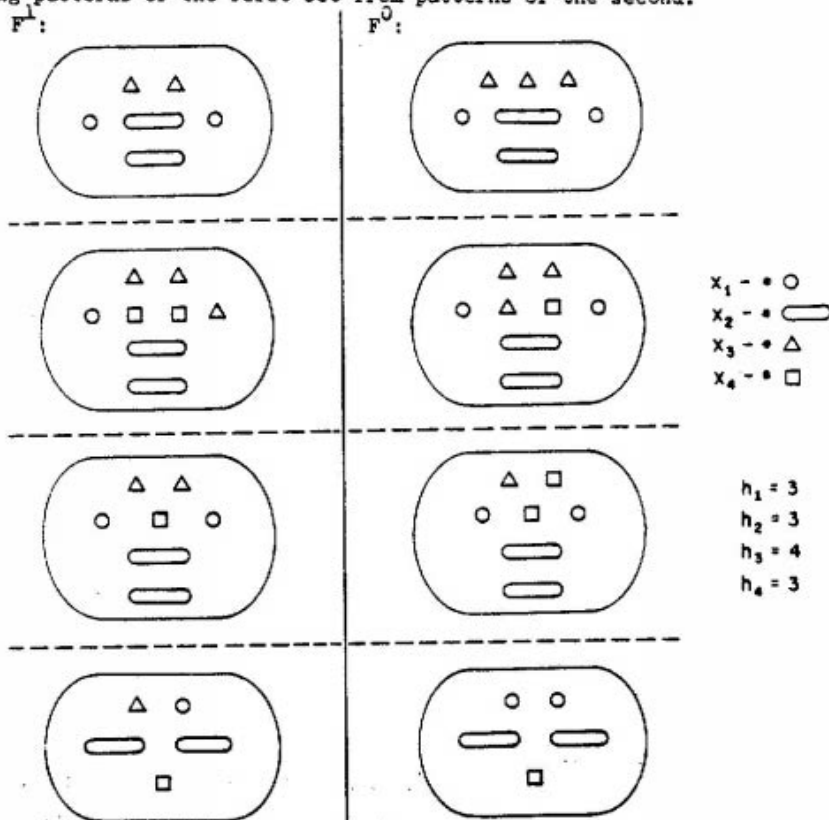


Figure 3

When a man tries to solve this problem, he usually looks for differences in the spatial relationships between elements in patterns of the two classes. In this case, however, there are no such differences, and therefore it seems to be difficult (for a man) to find a rule. On the other hand, for a computer program it is simpler, e.g. to count the numbers of elements of the same kind (e.g. circles, triangles, etc.) than to determine their relationships. In the example above, just such features happened to be relevant to the problem. The appropriate rule which involves these features can be found very easily using the VL approach. Assume that the numbers of circles, ellipses, triangles, and squares in the 'pictures' in  $F^1$  or  $F^0$  are elements of the input name sets  $H_1$ ,  $H_2$ ,  $H_3$  and  $H_4$ , respectively. Thus, to describe any 'picture' in  $F^1$  or  $F^0$  it is sufficient to assume  $H_1 = \{0,1,2\}$ ,  $H_2 = \{0,1,2\}$ ,  $H_3 = \{0,1,2,3\}$  and  $H_4 = \{0,1,2\}$ .

Sets  $F^1$  and  $F^0$  can now be represented by event sets  $\{(x_1, \dots, x_4)\}$ ,  $x_i \in H_i$ , and the problem becomes that of finding a cartesian cover of  $F^1$  against  $F^0$ . The result is shown in Fig. 4.

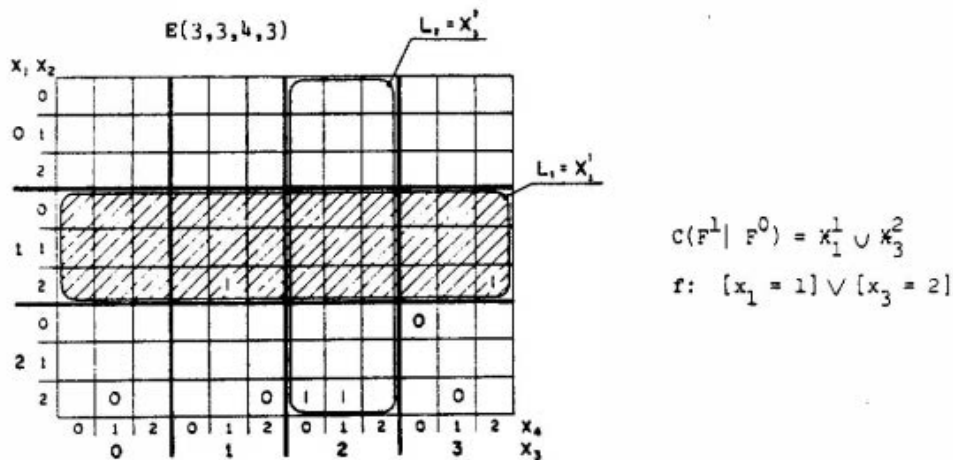


Figure 4

The  $VL_1$  formula which corresponds to the obtained cover is  $[x_1 = 1] \vee [x_3 = 2]$ , that is, it gives the rule: 'if a pattern has 1 circle or 2 triangles, then it belongs to the set  $F^1$ , otherwise to  $F^0$ .'

#### Example 2

Consider two classes of objects shown in Fig. 5.

If we would try to represent the objects of  $F^1$  and  $F^0$  by the events defined in example 1, i.e. by  $(x_1, \dots, x_4)$  (where  $x_1, x_2, \dots, x_4$  represent numbers of circles, ellipses, triangles, and squares, respectively), then events representing the first and second class would be identical. Thus, the above attributes are irrelevant for distinguishing the given 2 classes.

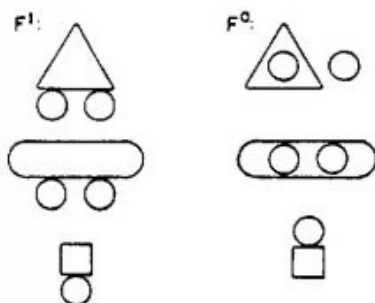


Figure 5

As a next step we can try checking to see if the binary relations between the elements in the pictures are distinguishing features of the two classes. Let us then introduce new variables  $x_5$ ,  $x_6$  and  $x_7$ ,

where:  $x_5$  and  $x_6$  represent the left and right elements, respectively, in a binary relation which relates the elements in a given picture. Their domains,  $H_5$  and  $H_6$ , are 'numerical names' of these elements:  $H_5 = H_6 = \{0,1,2,3,4\}$

where: 0 represents a left circle (if there are two)  
 1--a circle (or right circle)  
 2--a square,  
 3--an ellipse,  
 4--a triangle.

$x_7$  represents relations between the above elements. Its domain  $H_7$  consists of 'numerical names' of relations which are found in the pictures:  $H_7 = \{0,1,2,3\}$

where: 0 represents 'none of the relations is satisfied'  
 1--'on the left'  
 2--'contains '  
 3--'on top of.'

The class  $F^1$  is characterized by the set of events  $(x_5, x_6, x_7)$  in the universe  $E(5,5,4)$ :

$$F^1 = \{(4,0,3), (4,1,3), (3,0,3), (3,1,3), (2,1,3), (0,1,1)\}$$

and the class  $F^0$ :

$$F^0 = \{(4,0,2), (4,1,1), (0,1,1), (3,0,2), (3,1,2), (1,2,3)\}$$

(E.g. the event  $(4,0,3)$  states that a triangle is on top of a left circle.)

Event  $(0,1,1)$  appears in both classes and thus is not distinctive, so we remove it from these two sets.

$$\text{Sets } F^1 \text{ and } F^0 \text{ specify a function } f: E(5,5,4) \longrightarrow \{0,1,*\} \quad (31)$$

where  $\{e \mid f(e) = 1\} = F^1$  and  $\{e \mid f(e) = 0\} = F^0$ .



The minimal  $VL_1$  formula for  $f$  under the functional  $\langle t, g \rangle^*$ ,  
 where  $t$ --number of terms and  
 $g$ --the degree of generalization (as defined in 3.3),

$$\text{is: } [x_5 = 2:4][x_6 = 0,1][x_7 = 3] \quad (32)$$

The formula can be interpreted as: 'if a square, ellipse, or triangle is on top of one or two circles, then the picture belongs to  $F^1$ , otherwise to  $F^0$ .'

The minimal  $VL_1$  formula under the functional  $\langle t, s \rangle^{**}$ ,  $s$ --number of selectors (i.e. we seek a simpler description while permitting a greater generalization)

$$\text{is: } [x_6 = 0,1][x_7 = 3] \quad (33)$$

The formula says: 'if an object has any element (since  $x_5$  was dropped) on top of one or two circles, then it belongs to class  $F^1$ , otherwise to  $F^0$ .'

If we had found that binary relations were not distinctive, we could, in a similar fashion, apply ternary relations, etc.

### Example 3

Fig. 6 presents two classes of small 'pictures' which represent vectors  $(x_1, \dots, x_9)$ ,  $x_i \in H_i = \{0,1,2,3\}$ :

$x_7$	$x_8$	$x_9$
$x_6$	$x_1$	$x_2$
$x_5$	$x_4$	$x_3$

These vectors can be interpreted as events characterizing 2 classes of pictures (or any patterns). The  $x_i$  correspond to certain features which seem to be relevant in describing the pictures. Looking at these 'events' one can not observe any similarities between events in the same class and any distinctive features distinguishing the two classes.

The problem again is to deduce a simple classifying rule. Sets  $F^1$  and  $F^0$  define a function  $f: E(4,4,4,4,4,4,4,4,4) \rightarrow \{0,1,*\}$ ,  
 where  $(e \mid f(e) = 1) = F^1$  and  $(e \mid f(e) = 0) = F^0$ . (34)

We will seek the rule by synthesizing a minimal  $VL_1$  expression of the function  $f$  under  $\langle t, s \rangle$ .

Let us first find the interval  $VL_1$  formula (which directly corresponds to a cover  $I(F^1 \mid F^0)$ ). Evaluation of interval expressions is usually simpler than evaluation of non-interval expressions, since we only have to test if the values of variables lie between certain values rather than if they belong to certain

\* 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), but also that we require a minimal 'degree of generalization' resulting from these rules.

\*\* Assuming the functional  $\langle t, s \rangle$  means that we want to minimize the number of 'rules' as well as the number of variables in these rules, disregarding the 'degree of generalization' which may arise.

subsets (of course, the price for that is that the interval formulas are usually 'longer').

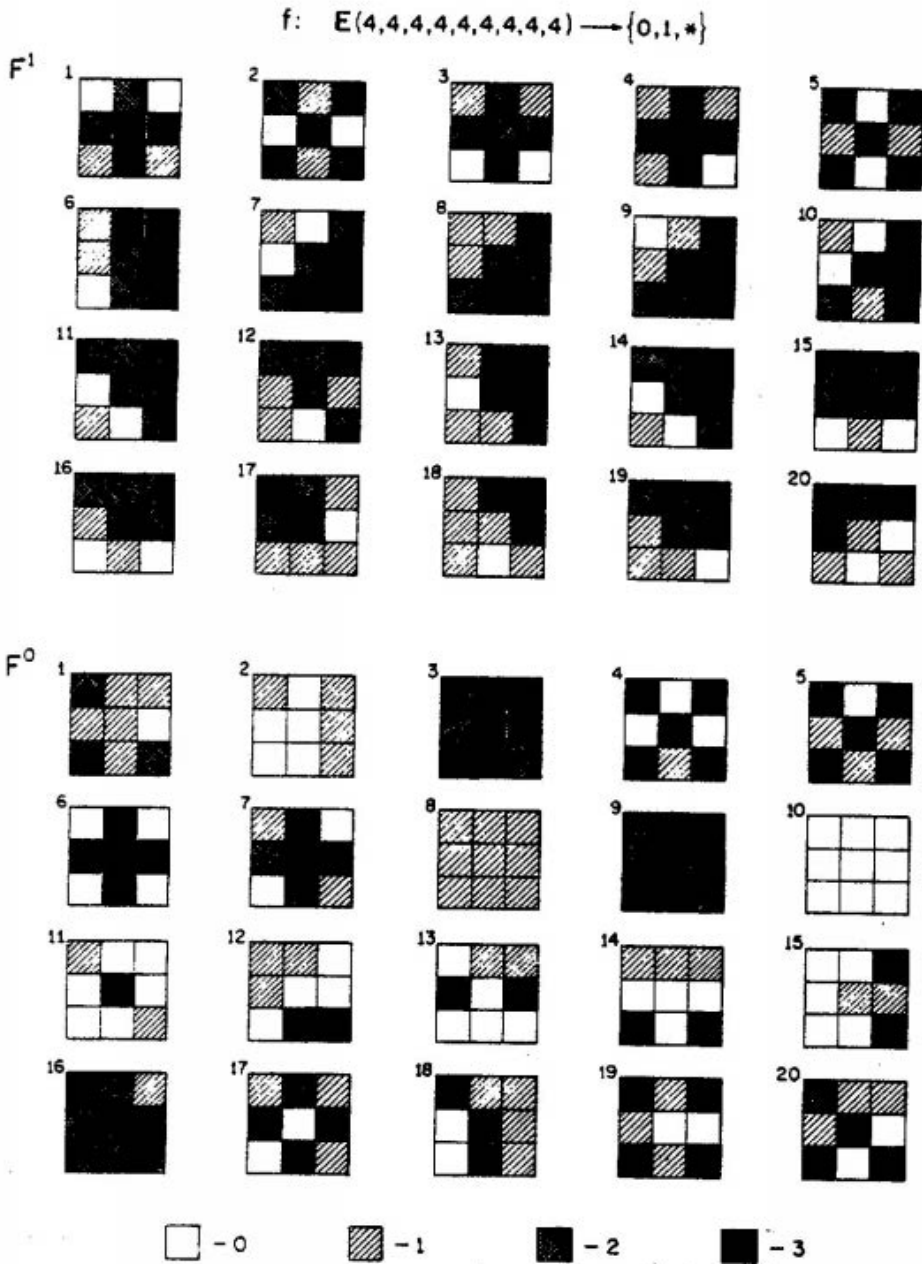


Figure 6

The first execution of the program AQ--31 produced an expression:

$$V_1 = [x_5 = 1][x_8 = 2:3] \vee [x_1 \neq 0][x_2 = 2:3][x_7 = 0:1][x_9 \neq 0] \vee \\ [x_1 = 2:3][x_2 \neq 0][x_4 = 0:1][x_9 = 1,2] \vee [x_1 = 3][x_2 \neq 3] \wedge \\ [x_3 \neq 3][x_9 = 2:3] \quad (35)$$

(We could have, of course, used any of the programs listed in section 3.3 since each program listed can principally do everything that all of its predecessors can.) This expression is illustrated in Fig. 7, where individual 'pictures' correspond to terms of the expression.

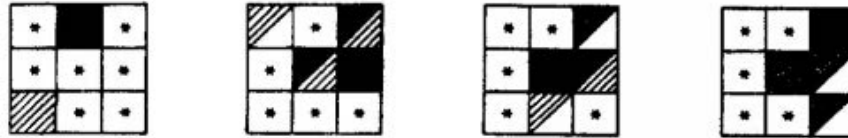



Figure 7

In Fig. 7,  means that the corresponding variable (in order to satisfy the appropriate selector) should have a value between 1 and 3 (the meaning of the other cells is analogous); \* denotes irrelevant variables.

$\Delta$  in this execution was 1 and the execution time  $\sim$  13 seconds.

The next execution (after certain rearrangements of the input event sets) gave an expression (Fig. 8):

$$V_2 = [x_2 = 2:3][x_5 = 1:2] \vee [x_1 = 2:3][x_4 = 1:2][x_8 = 1:3] \vee \\ [x_4 = 0][x_5 = 1:3][x_9 = 2:3]$$

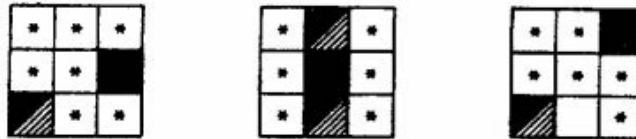



Figure 8

This expression only has three terms and in toto involves 6 variables ( $V_1$  involved 7). The value of  $\Delta$  was 0, and therefore this expression is minimal under  $\langle t \rangle$ , i.e. there does not exist an interval  $VL_1$  formula which would have less than 3 terms.

Finally, a non-interval formula (corresponding to a cartesian cover) found by the program AQ--4c was (Fig. 9):

$$V_3 = [x_2 \neq 1][x_5 = 1,2][x_9 \neq 1] \vee [x_1 = 2,3][x_2 \neq 3][x_5 \neq 2][x_6 \neq 0][x_9 \neq 0]$$

In Fig. 9, cell  means that the corresponding variable should have the value 0, 2 or 3 (the meaning of other cells is similar). Execution time (on an IBM 360/75) was 3.2 sec (the execution time in synthesizing cartesian covers is

usually considerably more than that for interval covers. In this case, however, it is less due to applying a special operation reduction technique called 'cutting' (Michalski, Tareski, 1972d)).



Figure 9

Thus, the rule for classifying objects of  $F^1$  and  $F^0$  is: 'if  $x_2 \neq 1$ ,  $x_5 = 1, 2$  and  $x_9 \neq 1$ , or  $x_1 = 2, 3$ ,  $x_2 \neq 3$ ,  $x_5 \neq 2$ ,  $x_6 \neq 0$  and  $x_9 \neq 0$ , then the object belongs to  $F^1$ , otherwise to  $F^0$ .'

It involves only two terms and in toto 5 variables (of the 9). In classifying objects the conditions expressed by these two terms can be checked in parallel, which would speed up the process (parallel computers, like ILLIAC III or ILLIAC IV, are therefore of special value for this kind of technique). If we were to test these conditions sequentially, then the average number of variables whose values have to be evaluated would be less than 5 (e.g. if we check first the first condition, which involves 3 variables, and an event satisfies it, then this event is classified without checking the second condition).

The rule thus found represents a generalization of the input sets  $F^1$  and  $F^0$  in the sense that events which did not occur in  $F^1$  and  $F^0$  will be classified by the rule. The classification of the new events may, of course, not be correct--which would mean that the rule is really not a 'true' rule for the classification. The rule is one of very many possible correct generalizations of the original data since it classifies the original events perfectly. If a new event would not be correctly classified, it would mean that the rule should be appropriately corrected (similar to how people correct hypotheses which explained old facts but failed when new facts were learned).

Another objection can be that in our system it may not be possible to express the 'true' rule (e.g. if the rule was, say,  $x_2^3 + x_4^2 > 9$ , since we do not have appropriate primitives in  $VL_1$ ). The system, nevertheless, would find an equivalent substitute (since any function (34) can be expressed in  $VL_1$ ).

This rule, of course, could be more 'complicated' than the 'true' one. (This is why in the Introduction we postulated having as many primitives as is feasible.)

#### Example 4

This example deals with the recognition of textures. Fig. 10 presents samples of 8 different classes of textures A, B, ... H. The problem is to determine a set of filters which would allow us to classify unknown textures into the

above classes (assuming, however, that the unknown textures are from these classes).



Figure 10

Fig. 11 presents samples of textures from the same classes in a digitized form (with a certain degree of noise and assuming 4 grey levels 0,1,2,3, which are represented in the same way as in Fig. 6.)

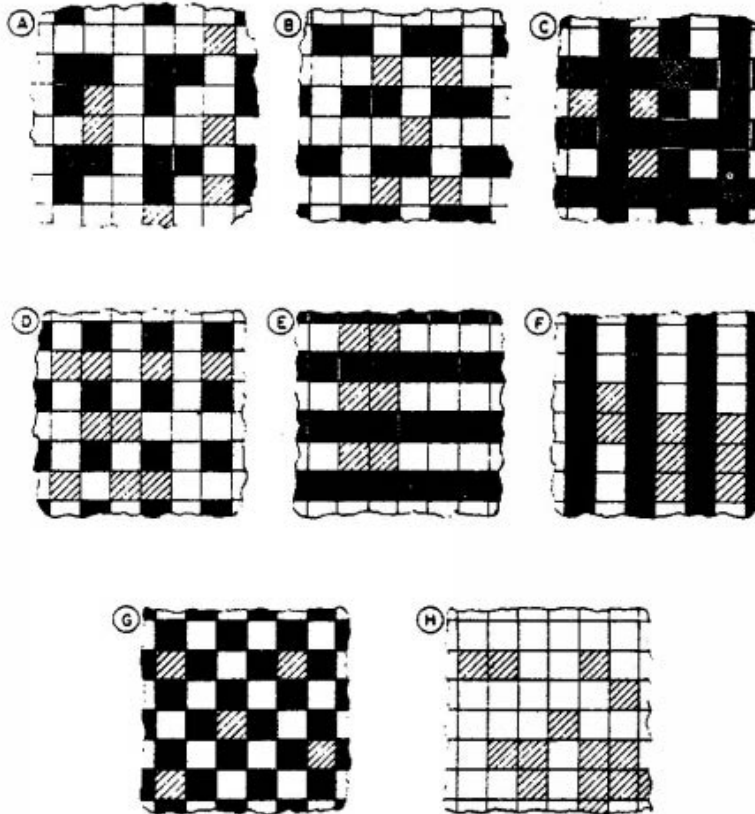


Figure 11

Assume that the output name set H consists of the 'numerical' names' which are assigned to the texture classes in the way shown in Fig. 12.

Let  $x_i$ ,  $i = 1, 2, \dots, 9$  represent grey levels of 9 neighboring elements in small  $3 \times 3$  squares extracted from the digitized textures:

$x_7$	$x_8$	$x_9$
$x_6$	$x_1$	$x_2$
$x_5$	$x_4$	$x_3$

Textures	M
(H)	0
(A)	1
(B)	2
(C)	3
(D)	4
(E)	5
(F)	6
(G)	7

Figure 12

Thus, the input name sets  $H_i$  are  $\{0,1,2,3\}$ . Let  $F^k = \{(x_1, x_2, x_3, \dots, x_9)\}$ ,  $k \in H = \{0,1,2,\dots,7\}$ , be a set of distinct events  $(x_1, x_2, \dots, x_9)$  which occur in the sample of the texture class, shown in Fig. 11, with the numerical name  $k$ . For example,  $F^1$ , the set of events extracted from texture class A, is:

$$F^1 = \{(2,0,0,1,3,3,0,0,0), (0,3,2,0,1,2,0,0,0), (3,3,0,2,0,0,0,0,0) \dots (3,0,1,0,2,3,0,0,1)\}$$

The first event in  $F^1$  represents the top left 3 x 3 square in the sample of the texture class A (Fig. 11), the second event - a similar square which is moved one cell to the right, and so

on, lexicographically, to the last event which represents the bottom right square. It can be verified that the sets  $F^k$ ,  $k=0,1,\dots,7$  are disjoint. The problem can now be formulated as that of synthesizing a minimal  $VL_1$  expression (under a suitable functional) of the function:

$$f: E(4,4,4,4,4,4,4,4,4) \longrightarrow \{0,1,2,\dots,7,*\} \quad (36)$$

where  $\{e \mid f(e) = k\} = F^k$ ,  $k=0,1,\dots,7$

In this example, however, accepting  $x_i$  as 4-valued is redundant in the sense that events with properly defined binary variables would also distinguishingly characterize the classes.

Transform the samples in Fig. 11 into binary samples by mapping the grey values 2,3 into 1 and 0,1 into 0. The events in  $F^k$  will become binary vectors  $(x_1^1, x_2^1, \dots, x_9^1)$ , where  $x_i^1 = \begin{cases} 1, & \text{if } x_i = 2,3 \\ 0, & \text{if } x_i = 0,1 \end{cases}$ . The new sets  $F^k$  are still disjoint, except for the event  $e = (0,1,0,0,0,1,0,0,0)$  which appears in the sets  $F^2$  and  $F^4$ . To make all sets disjoint, we remove  $e$  from  $F^2$  and  $F^4$  and create a set  $F^8 = \{e\}$  which can be interpreted as representing a new class 'B or D'.

Fig. 13 illustrates a minimal  $VL_1$  expression under  $\langle s,t \rangle$  for the VL function  $E \rightarrow H$  defined by the above sets  $F^k$ ,  $k=1,2,\dots,8$ , as found by AQUAL/1.

Individual 'pictures' correspond to terms in the formula (after returning to the old variables  $x_i$ ) and represent desired filters for the recognition of texture classes. E.g., if an event, extracted from an unknown texture, satisfies the filter in row B or D, then the texture is from the class B or D; if it does not satisfy the above filter, but satisfies one of the filters in row G, then the texture is from the class G; etc. (To distinguish between class B or D in the first case, another event for the texture should be extracted and compared with the filters.)

In order to increase the reliability of recognition, several events should be extracted from an unknown texture (in the extremum case--all events which occur) and compared with the filters. When contradictory recognition results are obtained, a majority rule can be applied.

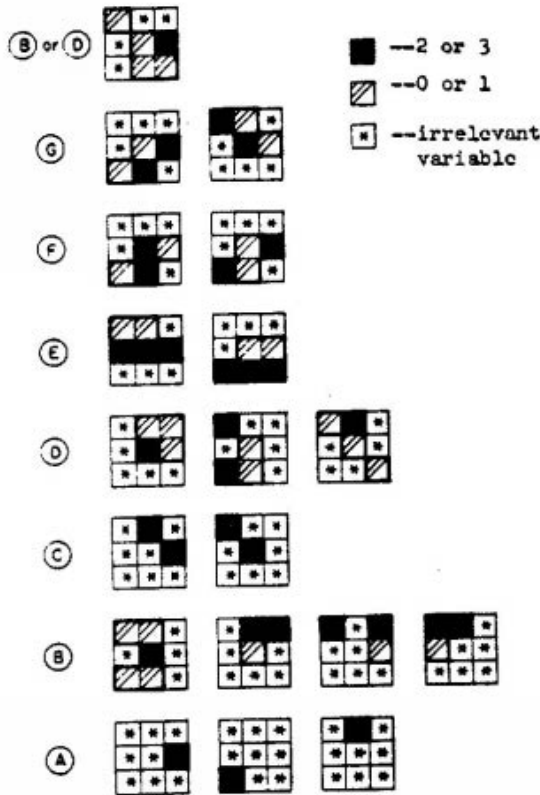


Figure 13

one set of filters, but a family of such sets may have to be constructed. The members of this family are then applied to an unknown texture in an iterative way (Michalski (1972a)).

The next example shows how  $VL_1$  is used in describing a complex graphical object. Such an object is treated as a structure of interrelated components (or parts). The components themselves can also be such structures, thus the same technique is applied in an iterative way till the descriptions involve only certain predefined elements.

Two types of descriptions are considered:

c-type -- which is viewed as an expression of a function

$$f_c: 2^C \longrightarrow R \tag{37}$$

where  $C$  is a set of components of the object ( $2^C$ --the power set of  $C$ )

$R$  -- a set of relations which relate subsets  $C_i \in 2^C$  of the object's components.

cr-type-- which is viewed as an expression of a function

$$f_r: 2^C \times R \longrightarrow D \tag{38}$$

In general, to recognize textures which are more 'complicated' than those considered in our example, a number of important operations should be performed prior to applying the described method. Among them are, e.g., proper adjustment of the resolution and the number of grey levels in picture quantization; construction of a ROC ('Receiver-operating-characteristic', as defined in statistical decision theory) for optimal selection of event size and distribution events into classes  $F^k$ ; etc. (Read and Jayaramamurthy (1972), Michalski (1972a)).

The events with which we describe textures do not have to be in the form which we used in the example. They can, of course, be of any suitable configuration. When the textures are complex, not only

where  $D$  is a set of truth-values representing the 'degree' to which a relation  $r \in R$  connects a subset of components  $C_i \in 2^C$ .

Here we will restrict ourselves to two-element subsets  $C_i \in 2^C$  (i.e. to binary relations) and  $D = \{0,1\}$ , where 1 means that a relation  $r \in R$  holds, and 0--does not hold.)

#### Example 5

Consider a line drawing in Fig. 14 representing a 'house'.

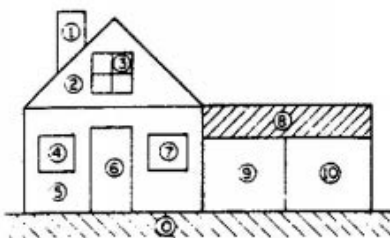


Figure 14

Assume, that using certain specialized procedures, an automatic recognition system is able to identify, in this line drawing, such elements as squares, rectangles, triangles, etc., determine their attributes (such as 'a vertical' (rectangle), 'a rectangle with a certain texture', etc.) and evaluate their relations (from a certain vocabulary of relations).

Our problem is how to use the above information to construct a possibly simple, easy to handle and logically irredundant description of the whole house. (This description should, e.g., make it easy to obtain answers to questions about the structure of the house.)


Assume that the components of the 'house' are numbered as shown in Fig. 14. The numbers (numerical names of the components) can be interpreted, e.g., as addresses of locations in a memory where the information about the attributes\* of components is stored. (In general, when components themselves are structures of interrelated subcomponents, then a description (not just the attributes) of the components should be stored. This description is constructed analogously to the description of the entire house.)

#### c-type description

Let  $x_1, x_2$  be variables whose domains, input name sets  $H_1, H_2$ , are sets of numerical names of the 'house' components:


$$H_1 = H_2 = \{0,1,2,\dots,10\} \quad (39)$$

where:

- 0 -- line with texture  below it
- 1 -- trapezoid
- 2 -- triangle
- 3 -- small square with cross

\*In the literature, attributes of elements are usually considered as unary relations and stored (handled) with other relations. We find such an approach rather inconvenient, except for special cases, especially when the components themselves are structures or have many attributes.



- 4 -- small square to the left of a vertical rectangle
- 5 -- large rectangle
- 6 -- vertical rectangle
- 7 -- small square to the right of vertical rectangle
- 8 -- horizontal rectangle with texture 
- 9 -- medium rectangle to the right of another medium rectangle
- 10 -- medium rectangle to the left of another medium rectangle.

Let the output name set  $H$  be a set of 'numerical names' of the binary relations found in the picture:

$$H = \{0,1,2,3,4\} \quad (40)$$

where

- 0 -- none of the relations holds
- 1 -- 'to the left of'
- 2 -- 'next to and the left of'
- 3 -- 'on top of'
- 4 -- 'contains'

Let  $F^k$ ,  $k=0,1,\dots,4$  be sets of events  $(x_1, x_2)$ ,  $x_1 \in H_1$ ,  $x_2 \in H_2$ , such that the value of  $x_1$  and  $x_2$  are components related in the 'house' by the relation  $k$ ,  $k \in H$ . (We will assume that relation '1' does not connect 2 objects where one of them is contained in a larger object which does not contain the second one).

$$\text{E.g. } F^3 = \{(1,2), (2,5), (5,0), (6,0), (8,9), (8,10), (9,0), (10,0)\}$$

Sets  $F^k$  define a function  $f_c: E \rightarrow H$ , where  $E = \{(x_1, x_2) \mid x_i \in H_i, i=1,2\}$ . A c-type description is a  $VL_1$  expression of  $f_c$ . A minimal  $VL_1$  formula under  $\langle t, s \rangle$  for  $f_c$  is:

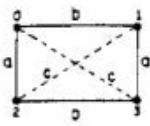
$$\begin{aligned} f_c = & 4[x_1=5][x_2=4,6,7] \vee 4[x_1=2][x_2=3] \vee 3[x_1=1][x_2=2] \\ & 3[x_1=2][x_2=5] \vee 3[x_1=5,6,9,10][x_2=0] \vee \\ & 3[x_1=8][x_2=9,10] \vee 2[x_1=5][x_2=8,9] \vee 2[x_1=9][x_2=10] \\ & 1[x_1=4,6][x_2=6,7] \vee 1[x_1=5][x_2=10] \end{aligned} \quad (41)$$

(If it is given that some relations are transitive (e.g. 'to the left of'), then the formula could be simplified even further.)

The first term of the formula states that component 5 ('large rectangle') is related to 4,6, and 7 (2 small squares and a vertical rectangle) by relation 4 (contains), etc. Using the formula, we can easily answer any question like: 'What relation connects components a and b?', 'Which components are related to component a by the relation r?', 'Which components are related by relation r?', etc.

E.g., a question 'Which relation connects components 2 ('triangle') and 5 ('large rectangle')' can be answered by inspecting terms of the expression, starting with the terms with the highest constants, in order to find the first term with selectors that are satisfied if  $x_1=2$  and  $x_2=5$ . The term  $3[x_1=2][x_2=5]$  is such a term and therefore the relation is 3 (on top of). Similarly, we can answer the other questions mentioned above.

The components of 'house' can also be considered as structures and described in a similar fashion. E.g., the component 5 ('large rectangle') (Fig. 15) can be described as follows.



Let  $x'_1, x'_2$  be variables with domains  $H'_1 = H'_2 = \{0,1,2,3\}$   
 where 0 represents vertex 0, 1 -- vertex 1,  
 2 -- vertex 2, 3 -- vertex 3  
 Let the output name set  $H$  be  $\{0,1,2,3\}$

Figure 15

- where 0 -- represents 'none of the relations hold'  
 1 -- represents the relation 'a segment of a straight line, of length 'a' between  $x'_1$  and  $x'_2$ ' where  $x'_1, x'_2 \in H'_1 = H'_2$   
 2 -- 'a segment of a straight line of length 'b' between  $x'_1$  and  $x'_2$ '  
 3 -- 'a segment of a straight line of length 'c', where  $c = \sqrt{a^2 + b^2}$ , between  $x'_1$  and  $x'_2$

(In order to give an attribute 'large' to the rectangle, 'a' and 'b' have to satisfy certain relations with regard to the size of the 'house'.)

A c-type description of the rectangle is a  $VL_1$  expression of a function

$$f_c: E \longrightarrow H, \text{ where } E = \{(x'_1, x'_2) \mid x'_1, x'_2 \in H'_1 = H'_2\}$$

A minimal  $VL_1$  formula under  $\langle ts \rangle$  for  $f_c$  is:

$$f_c: 3[x'_1=0,3][x'_2=0,3] \vee 3[x'_1=1,2][x'_2=1,2] \vee 2[x'_1=0,1][x'_2=0,1] \vee 2[x'_1=2,3][x'_2=2,3] \vee 1$$

In a similar way, other components of the 'house' can be described (using 'lower level' components or certain predefined elements).

cr-type description

Assume that  $x_1, x_2$  have the same meaning as in the c-type description.

Introduce a new variable  $x_3$  with domain  $H_3$  equal to the set previously defined as output name set  $H: H_3 = \{0,1,2,3,4\}$

As an output name set use the set  $H = \{0,1\}$ . Let  $F^1, F^0$  be sets of events  $(x_1, x_2, x_3), x_i \in H_i, i = 1,2,3$ :

$$F^1 = \{(x_1, x_2, x_3) \mid x_1 \text{ is related to } x_2 \text{ by the relation } x_3\}$$

$$F^0 = \{(x_1, x_2, x_3) \mid x_1 \text{ is not related to } x_2 \text{ by the relation } x_3\}.$$

Sets  $F^1$  and  $F^0$  define a function  $f_r: \{(x_1, x_2, x_3) \mid x_i \in H_i, i=1,2,3\} \longrightarrow H$ .

A cr-type description is a (minimal under  $\langle t,s \rangle$ )  $VL_1$  formula for  $f_r$  which is:

$$f_r: [x_1=5][x_2=4,6,7][x_3=4] \vee [x_1=2][x_2=3][x_3=4] \vee [x_1=1][x_2=2][x_3=3] \vee [x_1=2][x_2=5][x_3=3] \vee [x_1=5,6,9,10][x_2=0][x_3=3] \vee [x_1=8][x_2=9,10][x_3=3] \vee [x_1=5][x_2=8,9][x_3=2] \vee [x_1=9][x_2=10][x_3=2] \vee [x_1=4,6][x_2=6,7][x_3=1] \vee [x_1=5][x_2=10][x_3=1] \quad (42)$$

The c-type and cr-type descriptions of the 'house' are essentially equivalent. Computationally, the c-type is suited to a term by term evaluation of a formula, and cr-type to a parallel evaluation. In general, there are also other differences between the two types of descriptions with regard to applications (cr-type, e.g., permits the representation of 'weighted' relations, with 'degrees', i.e. where the set  $D$  has more than 2 elements).

The descriptions of objects constructed as illustrated in the above simple example can serve as models for these objects and be used for recognizing them in a broader context (such as a visual scene). In order to use them for recognition purposes, a number of new concepts (such as 'equivalence', 'isomorphism', 'homomorphism', 'homo-homomorphism' of  $VL_f$  formulas under a function  $f$ ), which go beyond the scope of the paper, are required.

#### Acknowledgment

The author expresses his appreciation to Professor Bruce H. McCormick for very helpful discussions, suggestions, and comments. The author would also like to thank Mrs. Star Starnes for the very accurate typing of the paper and Mr. Stanley Zundo for the excellent drawings.

#### 6. REFERENCES

- P. D. Givone, R. W. Snelsire (1968), The design of multiple-valued logic systems, Rep. Dept. of Elect. Eng. (June), State U. of New York at Buffalo, N.Y.
- R. S. Michalski (1969a), Synteza wyrazen minimalnych i rozpoznawanie symetrii funkcji logicznych, Rozprawa doktorska, Politechnika Śląska (an extended vers.: (1971), Prace Instytutu Automatyki PAN, 92, 1, Warszawa.
- R. S. Michalski (1969b), On the quasi-minimal solution of the general covering problem, Proc. V Intern. Symp. on Inform. Process. (FCIP69), Yugoslavia, Bled, Oct. 8-11, A3, 125.
- R. S. Michalski (1971a), A geometrical model for the synthesis of interval covers, Rep. 442, Dept. of Comp. Sc., U. of Illinois, Urbana, Ill.
- R. S. Michalski (1972a), Variable-valued logic: I--theory, II--computer implementation, III--applications, to appear as Rep. of Dept. of Comp. Sc., U. of Illinois, Urbana, Ill.
- R. S. Michalski, Z. Kulpa (1971b), A system of programs for the synthesis of combinational switching circuits using the method of disjoint stars, Proc. IFIP Congress, Ljubljana, Yugoslavia, Aug. 23-28, TA-2, 158.
- R. S. Michalski, B. H. McCormick (1971c), Interval generalization of switching theory, Proc. Third Annual Houston Conf. on Computer and System Sc., U. of Houston, April 26-27, 213 (an extended vers. in Rep. 442, Dept. of Comp. Sc., U. of Illinois, Urbana, Ill.).
- R. S. Michalski, B. H. McCormick (1972b), Cartesian Covers--a theoretical introduction, to appear as Rep. of Dept. of Comp. Sc., U. of Ill., Urbana, Ill.
- R. S. Michalski, P. Raulefs (1972c), Computer synthesis of cartesian covers, *ibid.*
- R. S. Michalski, Val Tareski (1972d), Interval Covers Synthesis--computer implementation of and experiments with algorithm  $A^Q$ , *ibid.*
- R. S. Nutter (1971), Function simplification techniques for Postian multi-valued logic systems, Dissertation, Graduate School, West Virginia University.
- J. S. Read, S. N. Jayaramamurthy (1972), Automatic Generation of Texture Feature Detectors, to appear in IEEE Trans. on C. (July).
- A. S. Wojcik (1971), Relationships between Post and Boolean algebras with application to multi-valued switching theory, Rep. R-512, Coordinated Science Lab, U. of Illinois, (June).
- Zhuravlev Yu. I. (1960) O nevozmozhnosti postroeniya minimalnykh normalnykh form funktsii algebry logiki v odnom klasse algoritmov, Doklady AN SSSR, Vol. 132, No. 3, 301.

## DISCUSSION

*Rosenfeld:*

You give a running time of 13 seconds for Example 3. Do you remember the times for Examples 4 (texture) and 5 (house)?

*Michalski:*

I didn't give the running time for Example 4 because the program was not completely finished when I was writing the paper. The complete version of AQVAL/1 which we presently have in operation is considerably faster than the previous version.

*Rosenfeld:*

How did you determine the set of properties or relations used in Example 5? Is this set of properties and relations a complete set by some criterion? If a machine had to systematically explore a space of possible properties and relations for that problem there might be some large combinatorial number of them. I may be misjudging, but it is my first reaction to the list of properties and relations given, that you hand-picked a set which had some relevance to reasonable structural descriptions. Is it possible that this set could have been automatically generated?

*Michalski:*

We would certainly like to be able to find out the relevant relations and eliminate irrelevant ones completely automatically. The present work is, of course, not a solution to all the problems. In this particular case we have chosen the most obvious attributes and relations. We have some ideas about how to attack the problem of finding a description by starting from original low level data and then producing higher level descriptions using the same approach in an iterative way.

*Rosenfeld:*

Then in fact let me comment that the texture examples are very nice in that nature hands you a simple method of picking an initial feature space, namely all the patterns that can occur in some small neighborhood. It would be nice if one could do such things for higher level problems such as the house. A second comment is: what you have here is potentially a very powerful grammatical inference procedure for finding simple structural descriptions. It might be enlightening to look at it from that point of view.

*McCormick:*

Michalski and myself have been in contact with Robert Langridge at Princeton on looking at protein structures and trying to detect a word structure in the sequence of amino acids. After all, proteins can be constructed from an alphabet of 21 letters (amino acids) plus the five discrete angles between adjacent amino acids. We intend to try to detect good words within available proteins. Using this technique which is in effect a grammatical inference procedure, we encounter a problem of more physical interest than many of those coming directly from the study of artificial languages.

*Rosenfeld:*

Solomon Marcus in Romania has also been trying to write grammars for DNA. I think he's trying to discover the grammar himself, but you have a powerful automatic tool for doing it.

*Nake:*

Could you say a word on how you extracted the events in the texture example; how do you scan, overlapping or not overlapping?

*Michalski:*

At the moment, we work in the following way: we first perform a number of pre-processing operations to discover, or to define, the resolution and number of grey-levels adequate for the given textures. I think that the resolution and number of grey-levels should not be assumed a priori but should be adopted to the given texture. Next, we extract sets of events which characterize our textures. I would like to emphasize that the tendency is to make the resolution and number of grey-levels as small as possible without losing considerably in practical validity. In the example with textures, binary representation happened to be good enough. Generally, we extract all possible events which occur in the textures. Then we find the optimal partition of events into classes, and apply AQVAL/1 to synthesize a proper VL expression.

*Nake:*

Will the parallel processing take care of the increasing computation time when you have finer resolution?

*Michalski:*

Yes, parallel computations are very important here. But, again, we check first if such a fine resolution is really needed in our concrete problem. Maybe we can decrease the resolution. This we check by constructing ROC's (receiver operating characteristics) as defined in statistical decision theory. We don't want to be killed computationally by not reducing the amount of information when it is possible.

*Nake:*

How do you actually represent the predicates and relations in terms of a computer language?

*Michalski:*

We use characteristic functions very successfully.

*Narasimhan:*

What slightly disturbs me about this work is that there seems to be an attempt here to convert what primarily are pictorial problems into non-pictorial combinatorial problems. The real crux of dealing with pictures is to be able to perform pictorial operations on pictures in order to abstract what you would call pragmatic information -- information about picture structures which need to be interpreted. Now, in all the examples you have discussed, you neatly skip this pictorial processing problem by presenting to your system something that is articulated in non-pictorial form. And then your system merely deals with this in a completely non-pictorial fashion. And even in the particular example where you probably do confront your system with a pictorial world, namely textures, your description of how the system works seems to imply that your system deals with this in a completely non-pictorial fashion. The whole approach seems to be quite appropriate in a pictorial system at some higher level, once one has left the pictorial domain. But the really open problems at the present stage lie precisely in the pictorial domain. We still do not know how to handle pictures in order to come up with relevant information.

*Michalski:*

It is my hypothesis, that at high levels of information processing our mind works in a similar way, whether we want to understand pictures or non-pictures. Our primary goal is to create an inference system which is able to discover the general rules independently of the form in which the information is given to it. Certainly this approach doesn't solve all image processing problems. We know, for example, that in our visual system we have a lot of built-in procedures which detect certain special properties -- line detectors, angle detectors, etc. So a way to attack the pre-processing problem is to have at our disposal a large number of different specialized procedures for various kinds of primitives. And then

the next level of processing can be performed by a general inference system of the kind that I described. I don't think it is possible that some general system will be able to do all this pre-processing. We can, however, apply our system to find filters for primitives like an angle, or a point, or a piece of line, etc.

*Narasimhan:*

I would just repeat the question that Rosenfeld asked before. The important thing is, given a set of feature filters, presumably your technique will tell you how to come up with a minimum description using these filters.

*Michalski:*

Yes, using primitives which have been detected in the picture.

*Narasimhan:*

But, you are not trying to tackle the problem of determining the vocabulary of primitives which will be appropriate for the class of pictures you are dealing with. You have a rather powerful procedure, in which it is possible to come up with certain kinds of minimal descriptions. The minimization is done on a certain vocabulary that is already given to the system. If, using this vocabulary, various equivalent descriptions can be produced, then your system can come up with a minimal description in some sense. But the question one is interested in asking is, whether the system can produce, for itself, an appropriate vocabulary.

*Michalski:*

My answer to this question is the following: human beings who have to operate in very different environments are able to measure very many different kinds of properties and primitives. They have built in or developed many different elementary concepts which they are using in describing the environment. Therefore, if we wanted to create an artificial man we would have to design all sorts of primitives which approximately would be the same as in human beings. But if we want to build a technical system and know to which kind of problems we want to apply it, we can certainly reduce the number of primitives which have to be at the disposal of the system. That simplifies the technical solution. We shouldn't demand that a system does everything for us. Problems as difficult as finding the set of elementary primitives should be done with the help of man, anyway at the present state of the game. But that doesn't mean that man has to provide really relevant primitives. He should provide a large number of primitives, as large as is computationally feasible. Our system will then discover which primitives and relations are relevant and useful.

*Hansen:*

Your program creates programs and -- noting that this is TC 2 which is interested in programming -- let me point out that the created program uses the "McCarthy or". It evaluates successive predicates until it gets one that is satisfied.

*Michalski:*

I didn't say what algorithm we were applying because there was not enough time. The algorithm is based on covering theory. To my best knowledge, I do not know any work which solves the problem in a similar way.