AQVAL/1--COMPUTER IMPLEMENTATION OF A VARIABLE-VALUED LOGIC SYSTEM VL1 AND EXAMPLES OF ITS APPLICATION TO PATTERN RECOGNITION

by

R. S. Michalski

Proceedings of the First International Joint Conference on Pattern Recognition, Washington, DC, pp. 3-17, October 30 - November 1, 1973.

acm

IFIP

OSA

Proceedings of the

FIRST INTERNATIONAL JOINT CONFERENCE ON PATTERN RECOGNITION

October 30 - November 1, 1973
Washington, D. C.

73 CHO 821-9C

AQVAL/1--COMPUTER IMPLEMENTATION OF A VARIABLE-VALUED LOGIC SYSTEM VL_1 AND EXAMPLES OF ITS APPLICATION TO PATTERN RECOGNITION

R. S. Michalski
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

Abstract

AQVAL/1 is a PL/1 program which synthesizes quasi-minimal formulas of the variable-valued logic[1] system VL_1 under cost (or complexity) functionals designed by a user from a set of available criteria. The paper presents a brief functional description of AQVAL/1 and reports current results of its application to two pattern recognition problems:

1. The synthesis of a classification rule for distinguishing between patients with cancer of the pancreas, primary cancer of the liver (primary hepatoma) and normal patients.

2. The design of spacial filters for non-uniform texture discrimination.

1. Introduction

The concept of a variable-valued logic system (VL system) was first introduced at the IFIP Working Conference on Graphic Languages in Vancouver, Canada, May 1972.[1] This concept assumes that every proposition (which conveys certain information or a description of a physical or abstract object) and all the variables in the proposition (used to represent any objects, e.g., other propositions) can assume their own number of 'truth-values' which are selected based on semantic and problem-oriented considerations. This assumption makes the system easily applicable to a variety of practical problems, in particular as a tool for describing objects or classes of objects. The assumption is supported by the observation that humans vary and adapt the 'degree of truth' or 'descriptive precision' of their statements according to the needs of the situation, and do so by changing the 'degree of truth' or 'quantization' of concepts represented by individual words or phrases in the statements. (E.g., when describing a person's height, different degrees of precision may be appropriate in various situations, so we may say that someone is 'tall', or is 'fairly tall' or is '6 feet 2 inches tall'. Each of these phrases implies the use of a different quantization of the concept 'height'.)

From the standpoint of formal logic, traditional binary logic systems and many-valued logic systems are special cases of a VL system.

Paper[1] defined and discussed a particular VL system, called VL_1. Important features of the VL_1 system are:

1. The formulas of the system (called VL_1 formulas) have very simple interpretation and can be very easily handled and evaluated (especially using parallel-sequential techniques).

2. The system is applicable for expressing and solving a variety of problems of discrete mathematics, especially problems which are intrinsically non-linear or involve variables which are measured on a nominal scale (i.e. when values of the variables are 'numerical names' of independent objects and, therefore, arithmetic relationships between these values have no meaning). The last feature makes the system applicable beyond the area of applicability of established pattern recognition methods, such as statistical techniques (parametric or nonparametric) or discriminant function methods.

The necessary condition for practical application of the system is the computational feasibility of its implementation, which means, in particular, the synthesis and minimization of VL_1 formulas.

A computer program, called* AQVAL/1, has been developed for the above tasks and the first trial applications of AQVAL/1 to various problems were quite satisfactory.

This paper gives a brief functional description of AQVAL/1 and presents current results from its application to two problems: the synthesis of classification rules for medical diagnosis, and the design of a set of spacial filters for non-uniform texture discrimination.

2. Definition of the VL_1 System

To describe AQVAL/1, knowledge of the VL_1 system has to be assumed. Therefore, to make the paper self-contained, we will review briefly the definition of VL_1 (as in paper[1] with slight modifications).

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

$$(X, Y, S, R_F, R_I) \tag{1}$$

where

X is a finite non-empty (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 $H_i = \{0,1,2,\ldots,\not{h}_i\}$, $i=1,2,\ldots,n$, $\not{h}_i$--a natural number.

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

$$H = \{0, 1, 2, \ldots, \not{h}\}, \quad \not{h}\text{--a natural number.}$$

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

S is the set of 13 improper symbols:

$$= \quad \neq \quad \vee \quad \wedge \quad \rightarrow \quad , \quad : \quad [\quad] \quad (\quad) \quad \leq \quad \geq$$

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

* The name AQVAL/1 was derived from 'Algorithm A^q applied for the synthesis of Variable-Valued Logic formulas.

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

2. A form $[L \# R]$

   where

   $\# \in \{=, \neq, \leq, \geq\}$ $\quad$ $R \in \{c, V_2\}$

   $L \in \{x_i, V_1\}$ $\quad$ $x_i \in X$

   c - a sequence of different non-negative integers separated by ',' or ':' and ordered by the relation $<$, or a name* of such a sequence;

   $V_1 V_2$ - wffs or names** of wffs,

   is a wff.

   The form $[L \# R]$ is called a <u>selector</u>. L and R, left and right parts of the <u>selector</u>, are called the <u>referee</u> (of L) and the <u>reference</u> (of R), respectively. If L is x_i, then L is called a <u>simple referee</u>. If R is c, then R is called a <u>simple reference</u>. A selector with a simple referee and a simple reference, i.e. a form $[x_i \# c]$, is called a <u>simple selector</u>.

   A simple reference, c, is called an <u>extended reference</u>, if it contains no ':'. If in an extended reference every maximal (under inclusion) sequence of consecutive integers of length at least three is replaced by a form $c_1{:}c_2$, where c_1 is the first and c_2 the last element of the sequence, then the obtained reference is called a <u>compressed reference</u>.

3. Forms (V), $\neg(V)$, $V_1 \wedge V_2$ (also written $V_1 V_2$) and $V_1 \vee V_2$, where V, V_1, V_2 -- wffs or names of wffs, are wffs.

   $\neg(V)$ is called the <u>complement</u> of V

   $V_1 V_2$ is called the <u>product</u> (or <u>conjunction</u>) of V_1 and V_2

   $V_1 \vee V_2$ is called the <u>sum</u> (or <u>disjunction</u>) of V_1 and V_2

R_I is a set of <u>interpretation</u> or <u>semantic rules</u> which assign to any wff V a <u>value</u> $v(V) \in H$, 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. $v([L \# R]) = \begin{cases} \not{h}, & \text{if } v(L) \# v(R) \\ 0, & \text{otherwise} \end{cases}$

   where

   $v(L)$, $L \in \{x_i, V_1\}$, is value of x_i, if L is x_i; otherwise $v(V_1)$, i.e. value of wff V_1

   $v(R)$, $R \in \{c, V_2\}$, is the sequence c, if R is c; otherwise $v(V_2)$, i.e. value of wff V_2

   $v(L) \# v(R)$, $\# \in \{=, \neq, \leq, \geq\}$, is true if $v(L)$ is in relation $\#$ with $v(R)$. If R is c, then:

   $v(L) = c$ $\quad$ ($v(L) \neq c$) is true if $v(L)$ is (is not) one of the integers in c, or is (is not) between any pair of integers in c separated by ':'

   $v(L) \leq c$ $\quad$ ($v(L) \geq c$) is true if $v(L)$ is smaller than or equal to (greater than or equal to) every integer in c (in normal use of these relations c will consist of just one element)

   If $v(L) \# v(R)$, then the selector $[L \# R]$ is said to be <u>satisfied</u>.

3. $v((V)) = v(V)$

   $v(\neg(V)) = \not{h} - v(V)$

   $v(V_1 V_2) \quad = \min(v(V_1), v(V_2))$

   $v(V_1 \vee V_2) = \max(v(V_1), v(V_2))$

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

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

$$4[x_1 = 0{:}4,7][x_2 \neq 0,5] \vee 2[x_3 \geq 3] \vee 1[x_5 = 0{:}4] \qquad (2)$$

The formula (2) is assigned the value 4 (briefly, <u>has value</u> 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 has value greater or equal to 3. 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.

In the sequel we will consider only a certain kind of VL_1 formulas, called <u>disjunctive simple</u> VL_1 formulas (DVL_1 formulas). A DVL_1 is a VL_1 formula which is a sum of simple terms, where each <u>simple term</u> is a product of a constant from H and one or more simple selectors. If, in a DVL_1, the references of simple selectors are in the form $c_1{:}c_2$ or c, then it is called an <u>interval</u> DVL_1 <u>formula</u>.

3. <u>Event Space and Cost Functionals for VL_1 Formulas</u>

The interpretation rules interpret VL_1 formulas as <u>expressions</u> of a function:

$$f: \; H_1 \times H_2 \times \ldots \times H_n \to H \qquad (3)$$

The set $H_1 \times H_2 \times \ldots \times H_n$, $H_i = \{0, 1, \ldots, \not{h}_i\}$, $i = 1, 2, \ldots, n$, includes all possible sequences of values of input variables and is called the <u>universe of events</u> or the <u>event space</u>. The event space is denoted by $E(h_1, h_2, \ldots, h_n)$ or, briefly, by E, where*
$h_i = c(H_i) = \not{h}_i + 1$ (i.e. $\not{h}_i = h_i - 1$, which is more mnemonic). The elements of E, vectors $(\dot{x}_1, \dot{x}_2, \ldots, \dot{x}_n)$, where $\dot{x}_i$ is a value of the variable x_i, $\dot{x}_i \in H_i$, are called <u>events</u> and denoted by e^j, $j = 0, 1, 2, \ldots, \not{h}$, where $\not{h} = h - 1$, $h = c(E) = h_1 h_2 \cdots h_n$. Thus, we can write:

$$E = E(h_1, h_2, \ldots, h_n) = \{(\dot{x}_1, \dot{x}_2, \ldots, \dot{x}_n) \mid \dot{x}_i \in H_i, i = 1, 2, \ldots, n\} = \{e^j\}_{j=0}^{\not{h}} \qquad (4)$$

It is assumed that values of the index j are given by a one-to-one function: $\gamma: \; E \to \{0, 1, \ldots, \not{h}\}$, specified by the expression:

$$j = \gamma(e) = x_n + \sum_{k=n-1}^{1} x_k \prod_{i=n}^{k+1} h_i \qquad (5)$$

---

* i.e. c denotes a variable whose values are such sequences.

** i.e. they denote variables whose values are wffs.

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

$\gamma(e)$ is called the number of the event e. For example, the number of the event $e = (2,3,1,4)$ in the space $E(5,4,2,5)$ is: $\gamma(3) = 4 + 1\cdot5 + 3\cdot2\cdot5 + 2\cdot4\cdot2\cdot5 = 119$.

Assuming that the domains of variables in the formula (2) have cardinalities: $c(H_1) = 8$, $c(H_2) = 6$, $c(H_3) = 7$, $c(H_4) = 2$, $c(H_5) = 5$ and the cardinality of H, $c(H) = 5$, the formula is interpreted as an expression of a function:

$$f: \; E(8,6,7,2,5) \to \{0,1,2,3,4\} \qquad (6)$$

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

$$f: \; E \to H \cup \{*\} \qquad (7)$$

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 (7) will be called variable-valued logic functions (VL functions).

In general, there can be a very large number of VL_1 formulas which express a given VL function (VL_1 expressions of f). Therefore, a problem arises of how to construct a formula which is minimal under an assumed cost (or complexity) functional. Depending on the application, different properties of VL_1 formulas may be desirable when expressing a given VL function, e.g., the minimal number of terms, of selectors, the minimal 'cost' of evaluation, etc. Thus, it is desirable that in a computer program which synthesizes minimal VL_1 formulas, there be available a number of cost functionals, from which the user may select the most appropriate for his type of application. For computational feasibility and convenience it is desirable to assume, however, that the available functionals be expressed in one standard form.

In the program AQVAL/1 (see Chapter 5) which synthesizes and minimizes disjunctive simple VL_1 formulas (DVL_1 formulas), a functional A measuring their minimality is assumed to be in the form:

$$A = \langle \text{a-list}, \tau\text{-list} \rangle \qquad (8)$$

where

a-list, called attribute (or criteria) 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 tolerance list, is a vector $\tau = (\tau_1, \tau_2, \ldots, \tau_\ell)$, where $0 \le \tau_i \le 1$, $i=1,2,\ldots,\ell$, 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:

$$A(V) \overset{\tau}{<} A(V_j) \qquad (9)$$

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,...--all irredundant DVL_1 expressions for f (a DVL_1 expression is called irredundant if removing any term or selector makes it no longer an expression for f).

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

$$A(V) \overset{\tau}{<} A(V_j) \text{ if } \begin{cases} a_1(V_j)-a_1(V) > T_1 \\ \text{or } 0 \le a_1(V_j)-a_1(V) \le T_1 \text{ and } a_2(V_j)-a_2(V) > T_2 \\ \text{or } \ldots\ldots\ldots \\ \vdots \\ \text{or } \ldots\ldots\ldots\ldots\ldots\ldots \text{ and } a_\ell(V_j)-a_\ell(V) \ge T_\ell \end{cases}$$

$T_i = \tau_i \cdot (a_{imax} - a_{imin})$, $i=1,2,\ldots,\ell$

$a_{imax} = \max_j\{a_i(V_j)\}$, $a_{imin} = \min_j\{a_i(V_j)\}$

Note that if $\tau = (0,0,\ldots,0)$ then $\overset{\tau}{<}$ denotes the lexicographic order in the usual sense. In this case, A will be specified just as A = <a-list>.

To specify a functional A one selects a set of attributes, puts them in the desirable order in the a-list, and sets values for tolerances in the τ-list.

4. Synthesis of VL_1 Formulas

4.1 Basic Concepts from Covering Theory

The synthesis of a minimal VL_1 formula for expressing a given function f under a functional A is based on the results of covering theory[2,3,4,5,6]. 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

$$L = \bigcap_{i \in I} X_i^{A_i} \qquad (10)$$

where

$$A_i \subseteq H_i, \; I \subseteq \{1,2,\ldots,n\} \qquad (11)$$

$$X_i^{A_i} = \{e = (\dot{x}_1, \dot{x}_2, \ldots, \dot{x}_i, \ldots, \dot{x}_n) \mid \dot{x}_i \in A_i\} \qquad (12)$$

'building blocks' which are used to express event sets. Event sets L are called cartesian complexes and sets $X_i^{A_i}$ --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

$$E^1 \subseteq \bigcup L_i \subseteq E \setminus E^0 \qquad (13)$$

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

$$\bigcap_{i \in I} {}^{a_i}x_i^{b_i} \qquad (14)$$

where ${}^{a_i}x_i^{b_i} = \{e = (\dot{x}_1, \dot{x}_2, \ldots, \dot{x}_i, \ldots \dot{x}_n) \mid a_i \le \dot{x}_i \le b_i\}$. A cover which consists of only interval complexes is called an interval cover. (In general, there can be many different types of complexes, e.g. complexes in which some of the literals are cartesian and some interval, the so-called cyclic complexes[4], etc.)

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 which expresses a function f: $E \to \{0,1,*\}$, assuming that $E^1 = \{e|f(e) = 1\}$ and $E^0 = \{e|f(e) = 0\}$. For example, if

$$C(E^1|E^0) = X_1^{1,3}\, X_3^{0,2,3} \cup X_2^{1,2,4} \text{ and } I(E^1|E^0) = {}^0X_1^2 \cup {}^1X_2^3\, X_4^0$$

then the corresponding VL_1 formulas are:

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

$$V_I = [x_1=0:2] \vee [x_2=1:3][x_4=0]$$

4.2 Synthesis of a Minimal DVL_1 Formula under a Functional A

A VL function f (7) can be specified by a family of event sets:

$$\{F^{\not{y}}, F^{\not{y}-1}, \ldots, F^0\} \qquad (15)$$

where $F^k = \{e|f(e) = k\}$, $k=\not{y},\not{y}-1,\ldots,0$. Let $M(E_1|E_2)$ be a minimal cartesian cover of E_1 against E_2 under functional A, defined as a cover such that the DVL_1 formula corresponding to it is minimal under A.

The synthesis of a minimal DVL_1 formula under A consists of the following steps:

1. Determine the minimal cartesian covers under A:

   $M^{\not{y}} = M(F^{\not{y}}|F^{\not{y}-1} \cup F^{\not{y}-2} \cup \ldots \cup F^0)$

   $M^{\not{y}-1} = M(F^{\not{y}-1}|F^{\not{y}-2} \cup F^{\not{y}-3} \cup \ldots \cup F^0)$

   $\vdots$

   $M^1 = M(F^1|F^0)$

2. Determine the products of selectors which are equivalent to complexes in the covers $M^{\not{y}}, M^{\not{y}-1}, \ldots, M^1$.

3. Multiply the products equivalent to complexes in $M^{\not{y}}$--by $\not{y}$, in $M^{\not{y}-1}$--by $\not{y}-1,\ldots$, in M^1 by 1.

4. The union of all terms thus obtained is a minimal VL_1 formula under A for the function f.

Thus, the synthesis of a minimal VL_1 is 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[7] 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 is to seek 'good' approximate solutions (except for simple, academic problems).

For the synthesis of cartesian covers the algorithm A^q ('quasi-minimal algorithm') can be applied[2,3]. This algorithm, based on the principle of disjoint stars, is a very simple and efficient method for the solution (generally sub-optimal) of any covering problem. It produces the so-called quasi-minimal cover, which is either minimal or approximately minimal under a given cost-functional. In the case when computations do not exceed the assumed memory and time restrictions, A^q produces also an estimate Δ of the maximal number of complexes in which the obtained and minimal covers can differ

$$c(M^q) - c(M) \le \Delta \qquad (16)$$

where M^q and M denote the quasi-minimal and a minimal cover, respectively.

5. AQVAL/1

AQVAL/1 is a PL/1 program which applies algorithm A^q to the synthesis of minimal DVL_1 formulas under a functional A. For the lack of space, we will not describe here the construction of AQVAL/1, but will give only its basic functional characteristics.

5.1 Specification of a VL Function

The VL function f, whose DVL_1 expression is to be minimized under a functional A, can be specified in the input data of the program in one of three forms:

1. By event sets: $F_e^{\not{y}}, F_e^{\not{y}-1}, \ldots, F_e^0$

   where: $F_e^k = \{e = (\dot{x}_1, \dot{x}_2, \ldots, \dot{x}_n) | f(e) = k\}$ (17)

   $k = \not{y}, \not{y}-1, \ldots, 0$

2. By sets similar to those in 1, but with events specified not as sequences of input variables, but by their numbers (5), i.e., values $\gamma(e)$:

   $$F_\gamma^k = \{\gamma(e) | f(e) = k\} \qquad (18)$$

3. By a DVL_1 formula (which can be already partially minimized; AQVAL/1 will try to minimize it further, if possible, with regard to functional A).

5.2 Specification of the Minimality Functional A

A user designs a functional A = <a-list, τ-list> by selecting a set of attributes from the available attributes, placing them in a desired order in the a-list, and setting up the τ-list. In the current version of the program the following seven attributes are available:

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 I is the set of indices of those variables (among $x_1, x_2, \ldots, x_n$, specified in the input data) which actually appear in the output VL_1 formula,

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

$$g(V) = \frac{1}{g_0} \sum_{k=1}^{\not{k}} \sum_{\ell=1}^{s_k} g(L_{k\ell}) \tag{19}$$

where

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

$L_{k\ell}$--the set of events which satisfy* the ℓ-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

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

5. $o(V)$ - the ordering criterion (not single-valued attribute). This criterion can be used when events in each set F^k, $k = \not{k}, \not{k}-1, \ldots, 0$ are assigned weight $w(e)$ (which represents, e.g., the frequency of event occurrence or 'importance' of the event) and a VL_1 formula is desirable, in which the first term T_1^k of each sequence of terms $(T_1^k, T_2^k, \ldots)$ with constant k, $k = \not{k}, \not{k}-1, \ldots, 1$ will have maximal possible weight $w(T_1^k)$, then the next term T_2^k will have the maximal possible weight, etc., where:

$$w(T_i^k) = \sum_{e \in E_i^k} w(e) \tag{20}$$

and E_i^k--the set of events from F^k, which satisfy T_i^k, but do not satisfy $T_{i-1}^k, T_{i-2}^k, \ldots, T_1^k$.
Formally, $o(V) = \langle -w(T_1), -w(T_2), \ldots \rangle$ and minimum $o(V)$ means that

$$o(V) \lessdot o(V_i) \tag{21}$$

where V_i--all other VL_1 formulas expressing f

$\lessdot$ the lexicographic order

6. $\ell(V)$ - the total length of references:

$$\ell(V) = \sum_{j=1}^{t} \ell(T_j) \tag{22}$$

where

$$\ell(T_j) = \sum_{i \in I_j} \ell(x_i)$$

I_j--the index set specifying variables occurring in selectors of the term T_j

$\ell(x_i)$--the number of constants in the compressed reference of the variable x_i, $i \in I$

t--the number of terms in V

7. $r(V)$ - relative scope of references:

$$r(V) = \sum_{j=1}^{t} r(T_j) \tag{23}$$

where

$$r(T_j) = \sum_{i \in I_j} \left(\frac{c_{max} - c_{min} + 1}{\ell_e(x_i)} \right)^2$$

c_{min}, c_{max}--minimal and maximal value in the reference of the variable x_i,

$\ell_e(x_i)$--the number of constants in the extended form of reference of variable x_i,

t, I_j--defined as previously.

5.3 Type of Variables

A simple selector $[x_i \# c]$, $\# \in \{=, \neq\}$, whose reference c is $c_1 : c_2$ or c, is called cyclic selector. A cyclic selector in which '#' is just '=' is called an interval selector.

If we want the VL_1 formula to be synthesized to include only cyclic (interval) selectors for a given variable x_i, then x_i is called (in the program) a cyclic (interval) variable, otherwise a cartesian variable.

AQVAL/1 permits a user to specify the type (cyclic, interval, or cartesian) of each input variable independently. Some guidelines for the specification of the type of the input variables follow.

Interval (and secondarily, cyclic) selectors are usually the simplest for evaluation (depending, in general, on the way the selectors are represented and on how the formula is evaluated--by a computer program or by a specialized machine). Also, the synthesis of minimal VL_1 formulas usually takes considerably less computational time and memory if variables are specified as interval (or secondarily, cyclic) rather than as cartesian variables. On the other hand, VL_1 formulas with interval (or cyclic) selectors will usually have considerably more terms and selectors than formulas obtained when no restrictions on the selectors were assumed.

If a variable takes values which have natural linear order (e.g., represent temperature, height, length, grey-level of a picture element, etc.) then it is usually advisable to specify it as an interval (or cyclic) variable. If this is not the case (e.g., the variable represents a set of independent objects, relations, properties, etc, i.e. is measured on a nominal scale), then the variable should be specified as a cartesian variable.-

5.4 Restriction Parameters <ms,cs>

Parameters <ms,cs>, called max-star and cut-star, are 'tree pruning' parameters which are used to control

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

the number of operations, and consequently, the execution time and memory requirements of the program. If a given problem is 'difficult' (i.e., n, h_i, h, $c(E^k)$, k = $\mathcal{K},\mathcal{K}$-1,...,0, are large), then <ms,cs> are set to small natural numbers (e.g., <5,5>, <1,1>, etc.), which causes a faster execution of the program and decreases memory requirements. The price for this is that the resulting VL_1 formula may be 'worse', i.e. further from the minimum with regard to the specified functional A.

A general explanation of the function of these parameters is as follows. The synthesis of a minimal VL_1 formula consists of a sequence of operations, each being a multistep process. In every step of the process, one set of complexes is transformed into another, usually larger. If the new set has more than ms elements, then it is reduced ('cut') to a smaller set with only cs elements, which are the 'best' candidates with regard to the functional A. This set is then the input for the next step of the process in which the parameters <ms,cs> are used again in the same way.

5.5 Mode: VL, DC,IC

AQVAL/1 permits a user to select among three kinds of VL_1 formulas to be synthesized:

'VL' - the formula is to be a DVL_1 formula without any restrictions.

'DC' - the formula is to be a DVL_1 formula with the property that the product of any two terms with different constants is 0 ('disjoint covers').

'IC' - the formula is to be a DVL_1 formula with the property that the product of any two terms with different constants may be non-empty, but must not be satisfied by any event from sets $F^{\mathcal{K}}$, $F^{\mathcal{K}-1}$,..., F^0 which are specified in the input data ('intersecting covers').

5.6 Size of the Program

The current version of the program (AQ-7) consists of about 860 PL/1 statements and the object module produced by the PL/1 (F) compiler requires memory capacity of about 60 k bytes (together with the PL/1 library routines--about 90 k bytes).

6. Examples of AQVAL/1 Application

We describe here some current results from the application of AQVAL/1 to two specific problems: (1) a medical diagnosis problem, and (2) a problem of non-uniform texture discrimination.

6.1 A Medical Diagnosis Problem

Medical diagnosis is concerned basically with determining a relationship between diseases and available information about patients, in particular, outcomes of medical tests. If the information is properly quantized, then the VL_1 system provides a way for attacking this problem. Namely, the system can be applied to determine the relationship from a set of known instances and express it in the form of VL_1 formulas.

The process of achieving this goal consists of two phases:

1. Learning phase

Based on a limited number of cases of the diseases under consideration and, for comparison, cases of 'normal health' (where by 'case' we mean a sequence of medical test results and other information about one patient) a VL_1 formula(s) is synthesized using AQVAL/1.

2. Testing phase

The obtained formula(s) is then used to diagnose some new cases of the diseases (and of normal health) in order to estimate its diagnostic performance.

To illustrate a medical diagnosis application of VL_1 we will report some results from experiements with data* characterizing cases of (1) cancer of the pancreas, (2) primary cancer of the liver (primary hepatoma), and (3) normal health.

The two above diseases are rare in the U.S.A. and are frequently misdiagnosed. The correct diagnosis of cancer of the pancreas was made, before operation or autopsy, in 50% of the cases examined in one study[9,10] and of primary cancer of the liver, ante mortem, in only 11% of the cases in another study[9,11]. Since the diseases are rare, there is little available statistical information about them, and various statistical classification techniques (using e.g., Bayesian rule or its modifications) are either not applicable or, if applicable, produce a very high error rate.

The experiments which we performed were of two types:

A. CANCER DETECTION: Determination of a classification rule for detecting patients with cancer, either of the pancreas or of the liver.

B. CANCER DISCRIMINATION: Determination of a classification rule for distinguishing between patients with cancer of the pancreas and patients with primary cancer of the liver.

The original data included values of 62 variables corresponding to general medical tests and parameters: (1-23)--admitting information, (24-43)--blood biochemistry and hematology tests, (44-62)--urinalysis, X-rays, and some other tests[9]. The data, however, were incomplete--for various patients different test outcomes were missing. The complete data, necessary for the learning phase (in the current version of AQVAL/1), were available, in a sufficient number of cases, only for some subsets of variables.

A. CANCER DETECTION

i. Learning phase

In these experiments, we used the following 19 variables:

* The data considered in our example were generously supplied by Dr. M. H. Loew of Purdue University (presently with Analytical Services Corporation, McLean, Virginia).

Admitting Information

x_3	- Age (years)	(h_3=7)

Blood Biochemistry & Hematology

x_{24}	- Sodium (meq/ℓ)	(h_{24}=4)
x_{25}	- Potassium (meq/ℓ)	(h_{25}=5)
x_{26}	- Chlorides (meq/ℓ)	(h_{26}=4)
x_{28}	- Fasting Glucose (mg/100 mℓ)	(h_{28}=7)
x_{29}	- Blood Urea Nitrogen (mg/100 mℓ)	(h_{29}=8)
x_{30}	- Serum Glutamic Oxaloacetic Transaminase (SGOT) (units)	(h_{30}=7)
x_{31}	- Alkaline Phosphatase (King-Armstrong units)	(h_{31}=6)
x_{34}	- Bilirubin: Total (mg/100 mℓ)	(h_{34}=6)
x_{36}	- White Cell Count (# per mm^3)	(h_{36}=6)
x_{38}	- Hematocrit (%)	(h_{38}=6)
x_{39}	- Neutrophiles (%)	(h_{39}=5)
x_{40}	- Eosinophiles (%)	(h_{40}=4)
x_{41}	- Basophiles (%)	(h_{41}=5)
x_{42}	- Lymphocytes (%)	(h_{42}=5)
x_{43}	- Monocytes (%)	(h_{43}=5)

Urinalysis

x_{45}	- Casts (count)	(h_{45}=4)
x_{46}	- WBC/hpf (count)	(h_{46}=5)
x_{47}	- RBC/hpf (count)	(h_{47}=5)

The ranges of variability of each of these variables were quantized into discrete units whose numbers (h_i) are shown on the right-hand side of the above list (the quantization was done according to the thresholds described in Loew[9], case '8'). Thus, the data characterizing each patient (i.e. each case) can be considered an event in the space $E = E(7,4,5,4,7,8,7,6,6,6,6,5,4,5,5,5,4,5,5)$. All variables were assumed to be interval.

The set of 87 events characterizing patients with either case of cancer ('cancer events') was partitioned into two sets:

F^c - learning set of 40 events,

T^c - testing set of 47 events.

The set of 63 events characterizing normal patients ('normal events') was partitioned into two sets:

F^n - learning set of 40 events,

T^n - testing set of 23 events.

The learning sets consisted only of completely specified events; the testing sets included completely and also incompletely specified events.

AQVAL/1 was applied to find two formulas: V(c/n) ('cancer versus normal') and V(n/c) ('normal versus cancer'), DVL_1 expressions of a function

$$f: E \to \{0,1\} \qquad (24)$$

which satisfies the conditions:

$$\{e/f(e) = 1\} = F^c \text{ and } \{e/f(e) = 0\} = F^n \text{ - for } V(c/n)$$

$$\{e/f(e) = 1\} = F^n \text{ and } \{e/f(e) = 0\} = F^c \text{ - for } V(n/c)$$

Such a pair of formulas, {V(c/n), V(n/c)}, is called a <u>conjugate pair of formulas</u> and is denoted by V(c*n).

Assuming the cost functional A = <(t,s,z)> (where $z(x_i)$, costs associated with variables x_i, were all assumed to be 1) and restriction parameters <ms,cs> = <15,5>, AQVAL/1 produced the following formulas:

$$1. \quad V(c/n) = [x_{38}=2:4][x_{42}=0][x_{43}\neq 0] \vee$$
$$[x_3=3:5][x_{26}\neq 3][x_{30}\geq 2][x_{38}\geq 3][x_{42}=1:3][x_{47}=0] \vee$$
$$[x_3\geq 5][x_{30}=1][x_{38}=3,4] \vee [x_3=2,3][x_{26}=3] \quad (25)$$

The formula comprises 4 terms, which consecutively cover (or are satisfied by) (28,28), (7,7), (6,3) and (2,2) events in the set F^c, respectively (the first number in each pair denotes the total number of events covered by a given term, and the second--the number of events covered by a given term and not covered by any of the previous terms). The formula depends in toto on only seven variables out of 19. The execution time on the IBM 360/75 was ≈ 39 seconds.

$$2. \quad V(n/c) = [x_{25}=1:3][x_{29}\leq 3][x_{30}\leq 4][x_{31}=1,2] \wedge$$
$$\wedge [x_{36}=2,3][x_{45}=0] \quad (26)$$

The formula comprises only one term (which covers all 40 events) and depends in toto on only six variables (out of 19). (Note that with one exception all variables in this formula are different from those used in the formula V(c/n), which is quite surprising.) The execution time was ≈ 22 seconds.

The fact that V(n/c) has only one term and is considerably simpler than V(c/n) (4 terms) seems to confirm the common intuition that cases of 'normal health' are more or less similar, i.e. all parameter values fall simultaneously into certain 'normal' ranges, unlike cancer cases which may be quite different because they may represent different types of cancer. (Note that in an interval DVL_1 formula each term corresponds to exactly one multidimensional interval in a subspace of E.)

Translating the VL_1 values of variables back into real values of tests (expressed in units given in the list of variables), the formula V(n/c) can be paraphrased:

$$[3< x_{25} \leq 6][x_{29} \leq 25][x_{30} \leq 90][5 < x_{31} \leq 25] \wedge$$
$$\wedge [5000 < x_{36} \leq 11000][x_{45} \leq 1] \quad (27)$$

(In the book[12] 'normal' ranges of the variables used in this formula are listed as: $4 \leq x_{25} \leq 5$; $9 \leq x_{29} \leq 20$; $10 \leq x_{30} \leq 40$; $4 \leq x_{31} \leq 13$; $4000 \leq x_{36} \leq 11000$; x_{45}--'a few casts may be present normally', page 164.)

2. Testing phase

The formulas V(c/n) and V(n/c) were applied individually and jointly (as a conjugate pair) to classify events in testing sets T^c and T^n.

In individual testing of the formulas we assumed that: V(c/n) produces a correct result when a 'cancer event' $e \in T^c$, is covered (i.e. satisfies the formula), or a 'normal event' $e \in T^n$, is not covered by the formula; V(n/c) produces a correct result when the opposite is true, i.e. an $e \in T^n$ is covered or an $e \in T^c$ is not covered.

The performance of these formulas tested individually is summarized in table 2a. Note that the total number of testing events was different for each formula (60 for $V(c/n)$ and 68 for $V(n/c)$, out of original 70). This is so because the table includes only results from testing events for which formulas were completely evaluated (as mentioned before, the testing sets included incompletely specified events for which it is not always possible to evaluate a formula).

It should be noted that to evaluate a given formula it is always sufficient to know only the values of the variables which actually appear in the formula (the number of which is usually much smaller than the totall number of variables constituting an event). Moreover, in many cases it is sufficient to know the values of only some of the variables appearing in the formula (e.g., it may be sufficient to know the values of variables occurring in only one term of the formula).

As can be seen from table 2, the formula $V(n/c)$ performed better than the formula $V(c/n)$, though both formulas seem to perform reasonably well ($V(c/n)$ — 83% (35 out of 42) and 89% (16 out of 18) correct diagnostic decisions for cancer and normal events, respectively; $V(n/c)$ — 89% (41 out of 46) and 96% (21 out of 22) correct decisions for cancer and normal events, respectively).

The 35 correctly classified cancer events were 'distributed' into (covered by) consecutive terms of $V(c/n)$: (28,28), (5,5), (4,2), (1,0), where the meaning of these numbers is the same as in the formula description.

Joint testing of two formulas (testing of a conjugate pair of formulas) involves judging the joint performance of two formulas for each testing event. Depending on the individual performance of each of the formulas, the joint performance has been categorized as shown on the left half of table 1 (detection categories). For a given event, each formula, taken separately, can give a correct diagnostic decision (denoted by 1 in the table), incorrect (denoted by 0) or no decision (denoted by *) -- if the event is not sufficiently specified. Thus for two formulas there are 9 possible pairs of outcomes, shown in the table. For the cancer detection results we accepted the rule that if any of the formulas detected a cancer, the joint decision was 'cancer', i.e. the unknown event was classified as a cancer event (otherwise as a normal event).

Thus, if one formula classified an event as a cancer event and the second-as a normal event, the event was classified as a cancer event. If this event was in fact a cancer event, then the joint performance was judged to be 'weakly correct'; otherwise (i.e. if it was a normal event) -- 'weakly incorrect'. The assignment of the other performance categories as shown in table 1 does not seem to require explanation.

The joint performance of $V(c/n)$ and $V(n/c)$ was summarized in table 2b. The joint performance of these formulas was correct for all 47 cancer testing events (100% correct) and for 20 of the 23 normal events (87% correct). Of the 3 normal events classified as cancer events, 2 were classified weakly incorrect, 1 - moderately incorrect and 0 - strongly incorrect.

B. CANCER DISCRIMINATION

1. Learning phase

In these experiments we tried to determine a rule, in the form of a DVL_1 formula, for distinguishing between cancer of the pancreas and primary cancer of the liver, based on 22 general medical tests. (These tests were selected from the original 62 variables for the practical reason of obtaining learning sets of a sufficient size.) The 22 variables (constituting events in the present experiments) included all of the 19 previously used variables (with x_{24} and x_{26} quantized differently) plus three additional variables:

Admitting Information

x_1	- Sex	$(h_1=2)$

Blood Biochemistry & Hematology

x_{24}	- Sodium (meq/ℓ)	$(h_{24}=7)$
x_{26}	- Chlorides (meq/ℓ)	$(h_{26}=6)$
x_{37}	- Hemoglobin (gm/100 mℓ)	$(h_{37}=5)$

Urinalysis

x_{44}	- Specific gravity (dimensionless)	$(h_{44}=5)$

The set of 38 cancer of the liver events ('liver events') was partitioned into two sets:

F^{ℓ} - learning set of 16 (completely specified) events,

T^{ℓ} - testing set of 22 (incompletely specified) events.

The set of 47 cancer of the pancreas events ('pancreas events') was partitioned into two sets:

F^p - learning set of 18 (completely specified) events,

T^p - testing set of 29 (incompletely specified) events.

AQVAL/1 was applied to find a conjugate pair of formulas $V(\ell*p) = \{V(\ell/p), V(p/\ell)\}$, where $V(\ell/p)$ and $V(p/\ell)$ are DVL_1 expressions of a function (24), satisfying conditions:

$$\{e/f(e) = 1\} = F^{\ell} \text{ and } \{e/f(e)=0\} = F^p \text{ - for } V(\ell/p)$$

$$\{e/f(e) = 1\} = F^p \text{ and } \{e/f(e)=0\} = F^{\ell} \text{ - for } V(p/\ell)$$

The cost functional was assumed to be, as before, $A = <(t,s,z)>$, and the restriction parameters $<ms,cs>$ were $<30,15>$.

The obtained formulas were:

$$1. \quad V(\ell/p) = [x_{24}=1:4][x_{28}=1:2][x_{29}\leq 4][x_{34}=2:4][x_{43}\geq 2] [x_3=2:5][x_{37}=1:3][x_{39}=3] \vee [x_{26}=1] \quad (28)$$

The formula comprises 3 terms, which consecutively cover (11,11), (5,4) and (1,1) events, respectively, in set F^{ℓ}. The formula depends on 9 variables (out of 22). The execution time was 61 seconds (this is a significantly longer time than in the previous examples due to the much larger restriction parameters).

2. $$V(p/c) = [x_1=0][x_{31}\leq 2][x_{36}=2{:}4][x_{44}\geq 3]$$
$$[x_{24}=2{:}4][x_{26}=3][x_{28}\geq 2][x_{38}=3]$$
$$[x_{28}=2{:}3][x_{30}=1{:}3][x_{34}=1{:}2] \qquad (29)$$

The formula comprises 3 terms, which consecutively cover (9,9), (5,5) and (5,4) events, respectively, in set F^p. The formula depends on 10 variables, 4 of which (x_{24}, x_{26}, x_{28}, x_{34}) were used in the formula $V(\ell/p)$. The execution time was 56 seconds.

2. Testing phase

The formulas $V(\ell/p)$ and $V(p/\ell)$ were applied individually and jointly to classify events in testing sets T^ℓ and T^p.

The performance of the formulas in both cases was summarized in table 3. Since the purpose of the formulas was to discriminate between two forms of cancer, the performance categories in joint testing were chosen slightly differently, as shown in the right half of table 1 (discrimination categories).

As can be observed in table 3, the performance of the formulas in individual and, as well, in joint testing was completely unsatisfactory (with the exception of classification of pancreas events by formula $V(\ell/p)$, which was 95% correct). Note also that the joint testing gave a very high percentage of indecisions (33%).

There can be few potential reasons explaining the results:

(1) the size of learning sets was too small for the program to make a proper generalization,

(2) the 22 variables used in the experiment, which are all general medical tests, may not have included any tests which are relevant in a sufficient degree to the distinction between cancer of the pancreas and cancer of the liver, thus ruling out the possibility of obtaining positive results in this experiment,

(3) the quantization of variables was not fine enough,

(4) the type of generalizations of input data, which AQVAL/1 is able to produce, is not adequate for the particular problem,

(5) the medical data which we used had errors (e.g., some cases listed in our data as cancer of the pancreas cases might have been in fact cases of cancer of the liver, or conversely).

1. With regard to reason 1 (learning sets too small), recall that the learning sets included only 16 (set F^ℓ) and 18 (set F^p) events, which makes it quite probable that they were of insufficient size for making adequate generalization. (To more clearly imagine the problem, note that the program was given only 16 + 18 = 34 specified events out of $\approx 8.5\cdot 10^{15}$ events constituting the event space.)

2. In view of the present poor results as compared to the previous quite satisfactory results (100% of cases of cancer (47 cases) were detected), reason 2 (irrelevant variables) seems to be more probable. Namely, in the cancer detection experiment, AQVAL/1 was able to produce good rules also based on relatively small learning sets -- 40 + 40 = 80 specified events out of $\approx 2.10^{14}$ (though $\approx$ 100 times larger, relative to the size of the event space, than in the cancer discrimination experiment). Thus, it seems to imply that the variables used in the experiments were sufficiently relevant for detecting the cancer but were not sufficiently relevant for discriminating between the two cancers.

3. It is natural to expect that differences in general tests, which were used in the experiment, will be much more subtle in distinguishing between two types of cancer than in distinguishing between cancer (either type) and 'normal health'. Therefore, for the cancer discrimination experiment a much finer quantization of variables might have been required. Since essentially the same quantization was used in the cancer detection and the cancer discrimination experiments, reason 3 (quantization of variables not fine enough) is perhaps a very significant factor here (new experiments using a finer quantization of variables are desirable).

4. Reason 4 (inadequate type of generalization) seems unlikely in view of the good results of the first experiment.

5. Reason 5 (incorrect data), of course, would undoubtedly cause bad results, if true.

Finally, it should be noted that the problem of cancer discrimination using general medical tests is not of very significant interest for medicine because there exists a specific test ('alpha-feto protein test') which can easily discriminate between these two cancers (once the presence of cancer has been detected). This imparts a greater value to the experiments on cancer detection than to those on cancer discrimination, when only general medical tests are used.

6.2 Texture Discrimination

This experiment deals with a problem of describing and discriminating non-uniform textures. AQVAL/1 together with some auxiliary programs has been applied to a solution of this problem. For the lack of space, we will give here only a very brief description of the general idea of the method applied and then illustrate it with an example (various aspects of earlier versions of this method were described in papers[1,13,14]).

Suppose there is given a set of texture classes $(\mathbf{T}^k)$, $k=0,1,2,\ldots,\mathcal{K}$, and the problem is to find an operator t, which applied to any texture $T^k \in \mathbf{T}^k$ produces the value $t(T^k) = k$. We will assume here that the $T^k \in \mathbf{T}^k$, $k=0,1,\ldots,\mathcal{K}$, represent digitized textures, namely $\alpha \times \beta$ matrices with entries, called picture elements, being positive integers between 0 and some number g, inclusively. We will also assume that the scale and position of the pictures of the real textures are 'appropriately' adjusted before the scanning and digitizing operations are applied (for learning and, as well, testing phases).

The operator t (a 'texture descriptor') is assumed here to be a composite operator:

$$t = (\Phi_1 \circ \Phi_2 \circ \ldots \circ \Phi_\mu) \circ d \qquad (30)$$

where $\Phi_i, i \in (1,2,\ldots,\mu)$ is called the filtering operator of the i-th iteration. Φ_i is a composite operator:

$$\Phi_i = a \circ \ell \circ p \circ v \qquad (31)$$

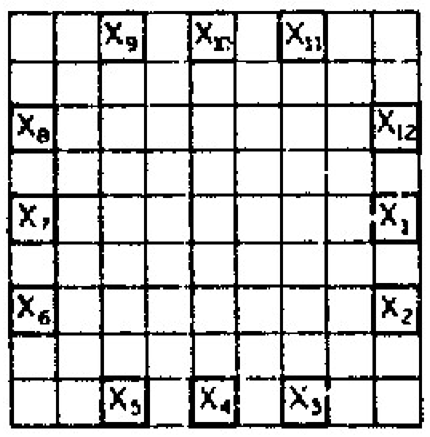
in which a, ℓ, p and v, generally dependent on i, are the following operators:

a - is an 'averaging operator', specified as a_{w_1,w_2}, w_1 and w_2 are divisors of α and β, respectively: $\alpha = \alpha_1 w_1$ and $\beta = \beta_1 w_1$, α_1 and β_1 -- some integers. The operator a applied to the matrix T^k (of size $\alpha \times \beta$) produces a new matrix T_1^k of the size $\alpha_1 \times \beta_1$, with entries equal to the average value of the submatrices ('windows') of size $w_1 \times w_2$ into which T^k is partitioned (of course, this operation on textures can be done by a scanner)

ℓ - is a 'grey-level reduction' operator, specified as $\ell_{\hat{g}}$, which applied to T_1^k produces a new matrix T_2^k of the same size, but with entries between 0 and $\hat{g}$, inclusively, rather than between 0 and g, $\hat{g} \leq g$.

p - is an 'event-shape' operator, specified as p_n, which applied to T_2^k produces a sequence of events $(\dot{x}_1,\dot{x}_2,\ldots,\dot{x}_n)$, $0 \leq x_i \leq \hat{g}$ (the sequence may include some events repeated many times).

The 'event-shape' operator p (and, as well, the operators a and ℓ) should be selected appropriately to the type of textures. Potentially this selection can be done by a computer program; in our experiments it was done with human assistance. In the example to be described, p was an operator, denoted as p_{12}, whose function can be described as follows. Consider a template:



with cut-out elements ('windows') which are marked $x_1,x_2,\ldots,x_{12}$ (assume that these windows are of the 'size' of elements in the matrix T_2^k). Suppose that this template is 'placed' on the matrix T_2^k (say, in the left top corner) and moved, step by step (where steps are of individual element size) through all possible positions in the matrix. In each position (j_1,j_2) an event $(\dot{x}_1,\dot{x}_2,\ldots,\dot{x}_{12})$ is extracted, where $\dot{x}_i$, i=1,2,...,12 is the entry of T_2^k 'seen' through the template window marked x_i. The sequence of events obtained at the end of this process, E^k, with the sequence of template positions, J^k, is defined to be the result of applying the operator p_{12} to T_2^k: $\langle E^k,J^k \rangle = p_{12}(T_2^k)$

v - is an operator which is a VL/1 formula expressing a function

$$f: \ \mathrm{E} = \{(\dot{x}_1,\dot{x}_2,\ldots,\dot{x}_n)\} \rightarrow \{0,1,2,\ldots,\not{p}\}$$

The result of applying v to $\langle E^k,J^k \rangle$ is defined as a matrix T_3^k such that its entry in position (j_1,j_2) is the value of the formula v for the event e extracted in position (j_1,j_2) of the template.

Let T(1) denote the result of applying the operator Φ_1 to an unknown texture T and T(i)--the result of applying the operator Φ_i to T(i-1), i=2,3,...,μ. T(i) is referred to as the transformed texture T after iteration i.

Operator d, applied to $T(\mu)$, selects a sample of $T(\mu)$ (of a size which is generally determined based on statistical considerations) and gives as the result the most frequently occurring value in this sample (in the case where more than one value occurs with the maximum frequency then other samples from $T(\mu)$ are tried).

The result of applying d is taken as the 'numerical name' of the class to which the unknown texture is classified.(the recognition decision).

Thus, the problem of describing and recognizing textures from some set of texture classes is considered here as the problem of designing an appropriate operator t. There can be in general, many operators t which can serve as recognizers (or descriptors) for given texture classes. A functional can be introduced for measuring cost (minimality) of these operators (which may include also the cost of estimated error rate) and the problem arises of constructing an operator t which minimizes the assumed cost functional.

We will give now a simple example of constructing an operator t and testing its performance. Consider the two pictures shown in figures 1 and 2 (Picture No. 15 and Picture No. 16) as two texture classes. As elements of these classes, consider various reasonably sized samples of these pictures. A problem is to determine an operator t which applied to any such sample of either picture gives the numerical name of the texture class from which this sample comes.

Pictures 15 and 16 were quantized into 128 x 128 matrices with g = 16 (16 grey-levels). Figures 3a and 4a show the result of applying to both pictures the operator $a_{3,3} \circ \ell_2$ (where ℓ_2 produced the binary pictures by thresholding the given matrix at the level of the mean grey-level value computed for the 'learning area'). The pictures in figure 3a and figure 4a were divided into two parts: a learning area - which served as a texture sample used to construct the filtering operators Φ_i and operator t, and a testing area - which served as a texture sample used to test the performance of the Φ_i and t.

The experiment consisted of three steps: (a) event extraction step, (b) AQVAL/1 step, (c) filtering and testing step. The results are summarized in table 4.

The sequences E^0 and E^1 of events $(\dot{x}_1,\dot{x}_2,\ldots,\dot{x}_{12})$, obtained after applying operator p_{12} to learning areas in figures 3a and 4a, are (in table 4a) represented by three sets of events:

$\hat{E}^0$ - of events which are in E^0 but not in E^1,

$\hat{E}^1$ - of events which are in E^1 but not in E^0,

$\hat{E}^\Phi$ - of events which are in both E^0 and E^1.

Columns $c(\hat{E}^1)$, $c(\hat{E}^0)$ and $c(\hat{E}^\Phi)$ give the length of these sequences. In constructing the filtering operators Φ_i, a cutting operator (considered as a part of the operator p) was applied to the sequences E^0 and E^1 which:

1. removed from E^1 and E^0 all vectors which occur in these sequences less than r times (r - occurrence threshold),

2. computed for the rest of vectors the parameter λ defined as:

$$\lambda(e) = \frac{e(E^1)}{e(E^1) + e(E^0)} \qquad (32)$$

where $e(E^1)$, $e(E^0)$ - number of times the event e occurs in E^1 and E^0, respectively.

3. removed from E^0 all events for which $\lambda > lb$, and from E^1 - all events for which $\lambda < ub$, where lb and ub are parameters called lambda lower bound and lambda upper bound, respectively, $0 < lb \leq ub < 1$.

4. produced sets F^0 and F^1 consisting of all distinct events remaining in sequences E^0 and E^1, respectively (note that due to the operation 3, F^0 and F^1 are disjoint). Columns $c(F^0)$ and $c(F^1)$ give the numbers of events in sets F^0 and F^1.

The experiment consisted of six computer runs (jobs):

1,2 - in which operators v were constructed as VL_1 formulas $V(F^0/F^1)$, which assume value 1, if an event $e \in F^0$ and value 0, if $e \in F^1$.

3,4,5,6 - in which operators v were constructed as formulas $V(F^1/F^0)$ which assume value 1, if $e \in F^1$, and value 0, if $e \in F^0$.

In table 4c by $V_{12}(F^0/F^1)$ is meant $V_1(F^0/F^1) \circ V_2(F^0/F^1)$, i.e., first is applied operator $V_1(F^0/F^1)$ and then $V_2(F^0/F^1)$; ($V_{12}(F^1/F^0)$, $V_{123}(F^1/F^0)$, etc., have analogous meaning.)

Parameters cl and ct (table 4c) called correctness ratio for learning area and correctness ratio for testing area, respectively, are defined:

$$cl = \frac{CL}{TL} \qquad ct = \frac{CT}{TT}$$

where CL and CT are total numbers of events which VL_1 formula classified correctly (i.e., produced for them value 1, if $e \in E^1$ and value 0 if $e \in E^0$), in learning and testing areas, respectively. TL and TT are total number of entries in learning and testing areas, respectively.

Figures 3b,c and Figures 4b,c show the transformed textures of 1st and 2nd iteration, i.e., results of applying operators $\Phi_i = a \circ f \circ p \circ v$, i=1,2, to pictures in figure 15 and figure 16 where a, f, p and v are defined in table 1 (jobs 1 and 2). As we can see, the densities of 'Os' in figure 3c and '1s' in figure 4c are very large, both in learning and testing areas (they are measured by cl and ct). Thus, the operator d will give the correct recognition decisions even if the samples taken from pictures in figures 3c and 4c are relatively very small (recall that the decision will be correct if: the number of 'Os' is greater than the number of '1s' in a sample taken from the testing area in figure 3c or, if the number of '1s' is greater than the number of 'Os' in a sample taken from the testing area in figure 4c).

7. Concluding Remarks

Let us briefly summarize the most important properties of the VL_1 system and its AQVAL/1 implementation, from the application viewpoint:

1. AQVAL/1 logically infers quasi-minimal VL_1 formulas (under an assumed cost-functional) from sets of events representing 'facts' or 'true' statements about any objects or classes of objects. This process is problem-independent; it is based entirely on logical properties of the VL_1 system.

2. The VL_1 formulas inferred by AQVAL/1 are expressions of a function

$$f: H_1 \times H_2 \times \ldots \times H_n \rightarrow H$$

where H_i, i=1,2,...,n and H are f.n. sets of non-negative integers.

An important property of the VL_1 system is that every such function can be expressed by a VL_1 formula. Since a large number of problems from different fields, in particular various pattern recognition problems, can be formulated as problems of finding an expression for such a function, AQVAL/1 has a potential for broad application.

3. The system can be especially useful for solving problems which are intrinsically non-linear or involve variables measured on nominal scale (i.e. values of variables denote independent objects and arithmetic relationships between the values have no meaning; such variables are specified in the program as cartesian variables). This is a very unique feature of the system.

4. The VL_1 formulas are very simple for interpretation and evaluation. (When only interval variables are used, they correspond to decision rules involving only hiperplanes perpendicular to individual variable axes.)

5. AQVAL/1 automatically reduces redundant variables.

Acknowledgment

The texture experiments reported in the paper were supported by contract AT(11-1)-2118 with the U.S. Atomic Energy Commission. The author gratefully acknowledges the programming help he received from the following people at the various stages of the development of AQVAL/1 and its earlier versions: Val Tareski (AQ-1,2,3), Peter Raulefs (AQ-4,5,6), Celso Frazer and Kurt Hirchert (AQVAL/1 --v7). The author expresses sincere thanks to his research assistants: Mr. Lee Chapman for help in editing the paper and in the experiments with medical data, and Mr. Tze-Wah Wong for help in the experiments concerning the method of texture discrimination. Thanks go also to Mr. S. N. Jayaramamurthy, who provided the event extraction program used in the experiments with textures.

List of 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, Vancouver, Canada, May 1972.

[2] 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.

[3] Michalski, R. S., "Synthesis of Minimal Structures of Combinational Automata and Recognition of Symmetry in Switching Functions," Proceedings of the Institute of Automatic Control, No. 92, Polish Academy of Sciences, Warsaw, 1971, 155 pp. (in Polish).

[4] 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.

[5] Michalski, R. S., "A Geometrical Model for the Synthesis of Interval Covers," Report No. 461, Department of Computer Science, University of Illinois, Urbana, Illinois, June 24, 1971.

[6] Michalski, R. S. and McCormick, B. H., "Cartesian Covers--A Class of Expressions for Finite Discrete Sets" (to appear).

[7] Zhuravlev, Yu. I., "O nevozmozhnosti postroeniya minimalnykh normalnykh form funktsii algebry logiki v odnom klasse algoritmov, Doklady AN SSSR, Vol. 132, No. 3, p. 301.

[8] Michalski, R. S., "Variable-Valued Logic System VL_1: I Elements of Theory, II Computer Implementation, III Examples of Application (to appear).

[9] Loew, M. H. and Fu, K. S., "Computer-Aided Medical Diagnosis Using Sequential Pattern Recognition Techniques," Purdue University, School of Electrical Engineering, TR-EE 72-14, May 1972.

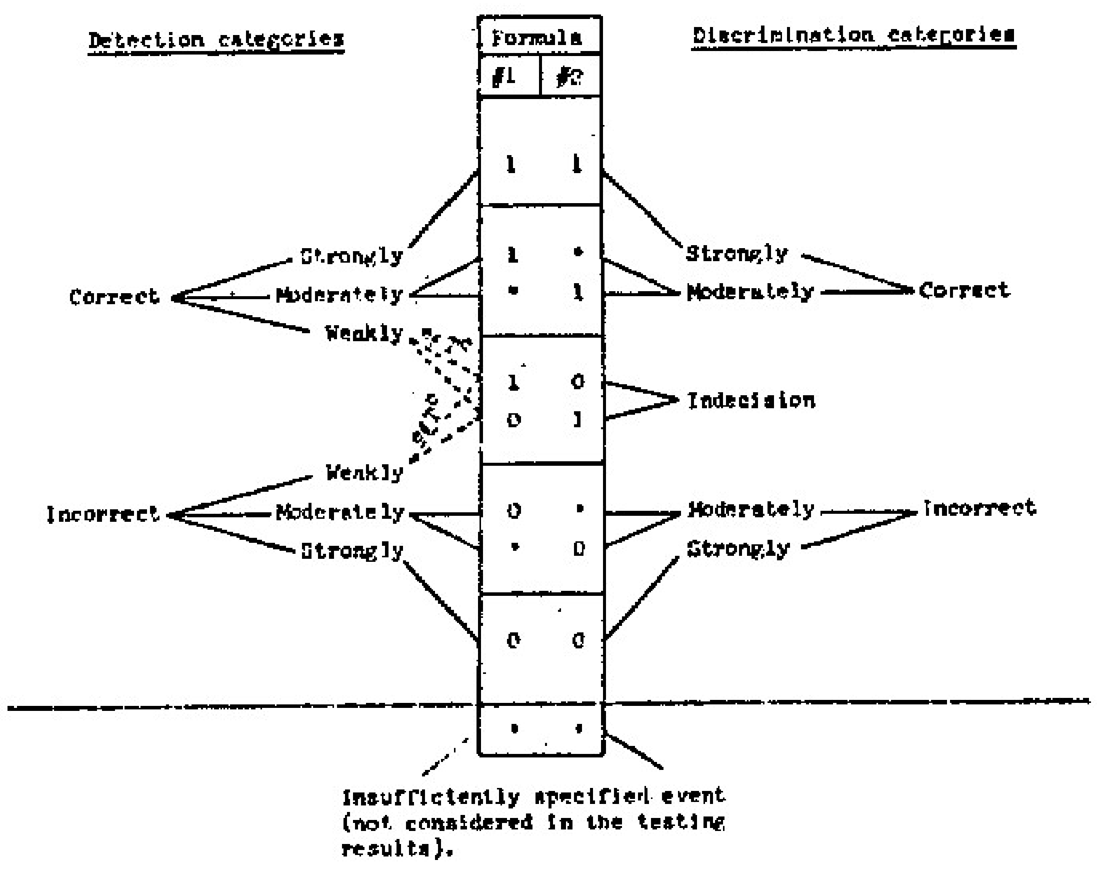
[10] Rastogi, H. and Brown, C. H., "Carcinoma of the Pancreas," Cleveland Clinic Quarterly, Vol. 34 October 1967, pp. 243-263.

[11] Schwartz, S. I., "Surgery of the Liver," in Diseases of the Liver W. T. Foulk, ed. New York: McGraw-Hill, 1968, pp. 123-136.

[12] Garb. S., "Laboratory tests in common use." SPRINGER Publish. Co., New York, N.Y. 1971.

[13] Read, J. S. and Jayaramamurthy, S. N., "Automatic Generation of Texture Feature Detectors," IEEE Transactions on Computers, July 1972.

[14] McCormick, B. H. and Jayaramamurthy, S. N., "Analysis of Texture," Report No. 531, Department of Computer Science, University of Illinois, Urbana, Illinois, July 1972.



PERFORMANCE CATEGORIES FOR A CONJUGATE PAIR OF FORMULAS

Table 1.

		V(c/n)			V(n/c)		
		Correct	Incorrect	Total	Correct	Incorrect	Total
Cancer	# cases	[illegible]	7	[illegible]	[illegible]	5	[illegible]
	Percentage	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Normal	# cases	10	[illegible]	[illegible]	[illegible]	1	[illegible]
	Percentage	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Total	# cases	[illegible]		[illegible]	[illegible]	[illegible]	[illegible]
	Percentage	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Individual testing of formulas V(c/n) and V(n/c).

V(c+n) = {V(c/n), V(n/c)}								
Correct				Incorrect				Total
Strongly	Moderately	Weakly	Total	Weakly	Moderately	Strongly	Total	
[illegible]	[illegible]	[illegible]	[illegible]		0	0	0	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]		[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	5		[illegible]	2	1	[illegible]	3	[illegible]
[illegible]	[illegible]		[illegible]	[illegible]	[illegible]	[illegible]	13%	[illegible]
[illegible]	11	[illegible]	[illegible]	2	1	[illegible]	3	[illegible]
63%	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Joint testing of formulas V(c/n) and V(n/c).

t.

CANCER DETECTION

Table 2.

		V(l/p)			V(p/l)		
		Correct	Incorrect	Total	Correct	Incorrect	Total
Liver	# cases	[illegible]	[illegible]	11	10	11	21
	Percentage	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Pancreas	# cases	[illegible]	1	[illegible]	[illegible]	[illegible]	[illegible]
	Percentage	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Total	# cases	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	Percentage	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Individual testing of formulas V(l/p) and V(p/l).

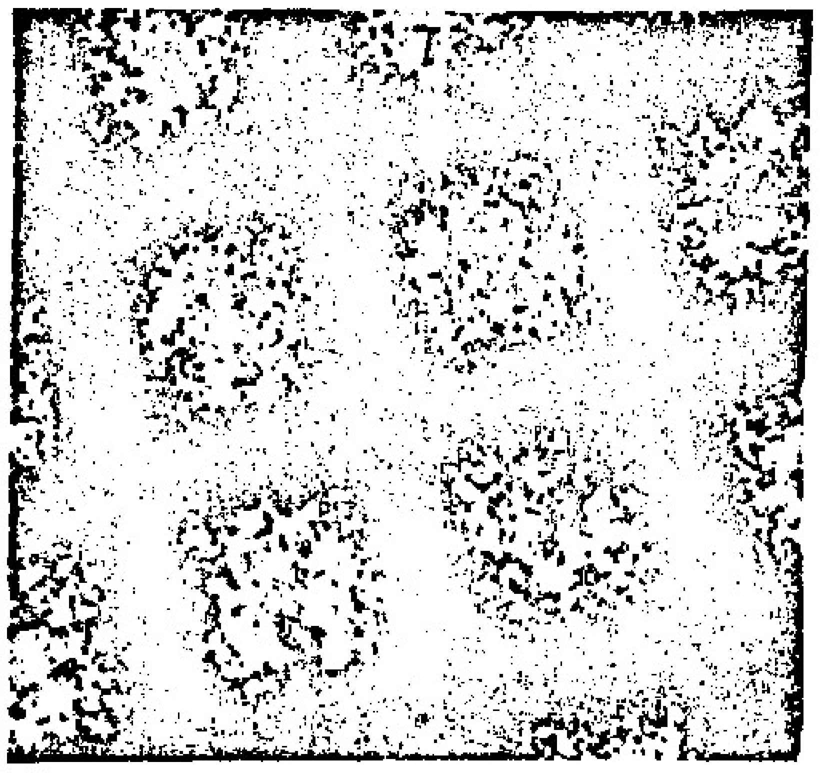
V(l+p) = {V(l/p), V(p/l)}							
Correct			Indecision	Incorrect			Total
Strongly	Moderately	Total		Moderately	Strongly	Total	
3	9	[illegible]	3	[illegible]	[illegible]	9	[illegible]
14%	9%	[illegible]	14%	14%	14%	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Joint testing of formulas V(l/p) and V(p/l).

h.

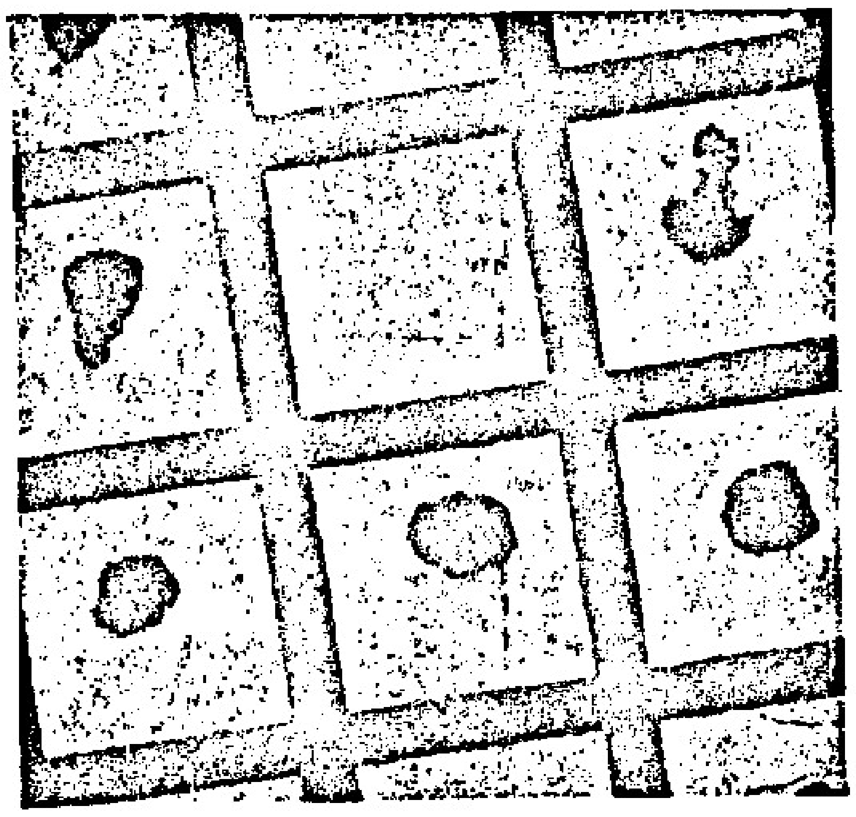
CANCER DISCRIMINATION

Table 3.



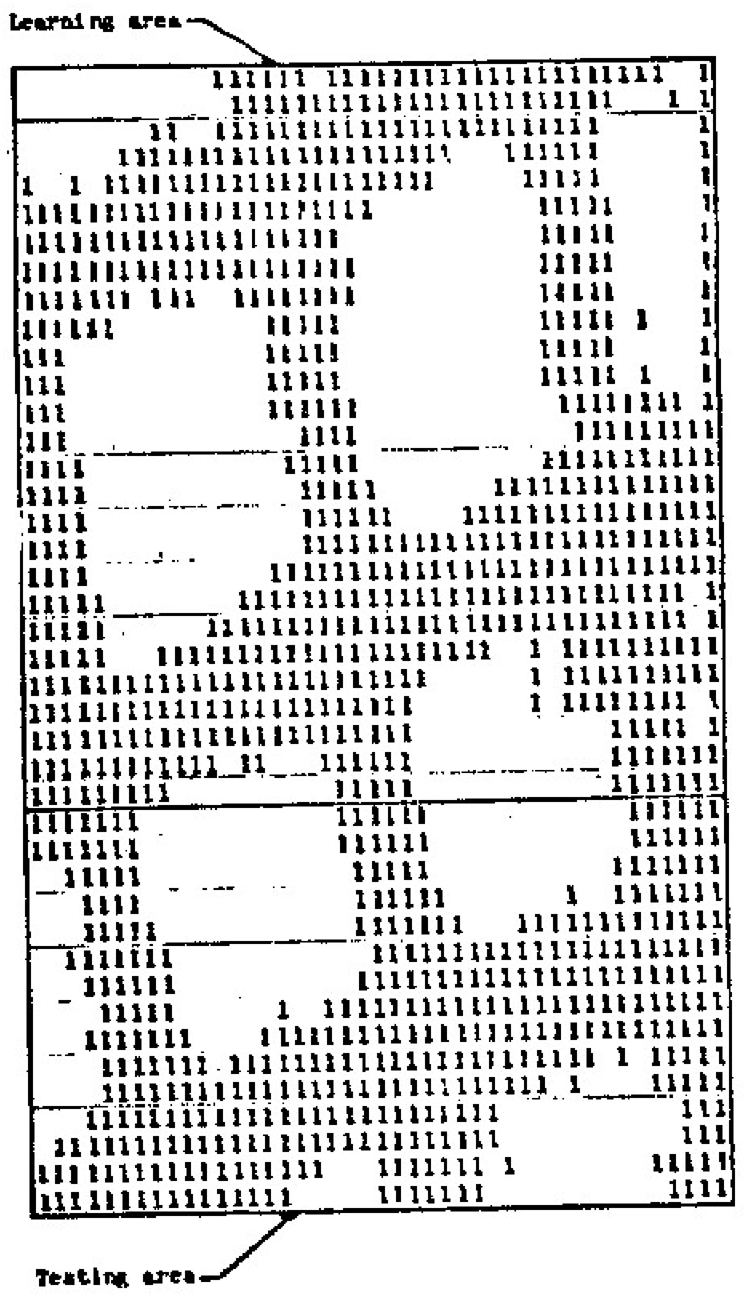
Picture No. 15

Figure 1



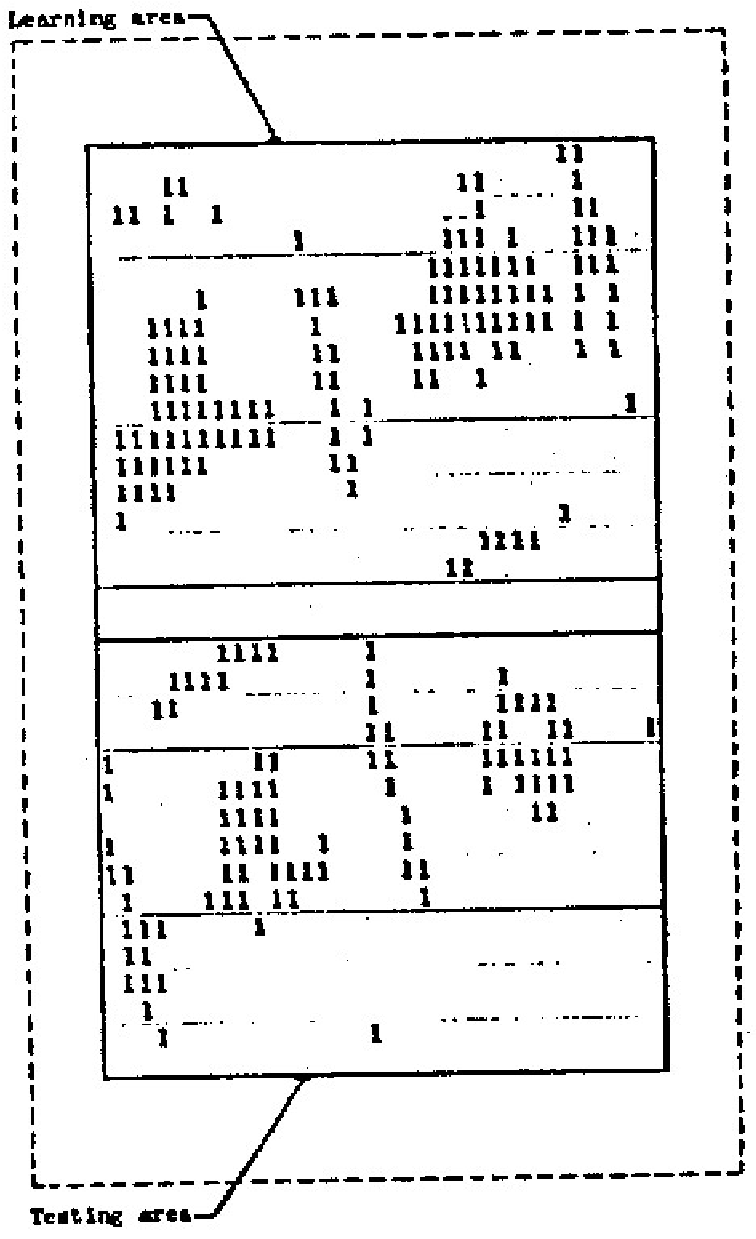
Picture No. 16

Figure 2



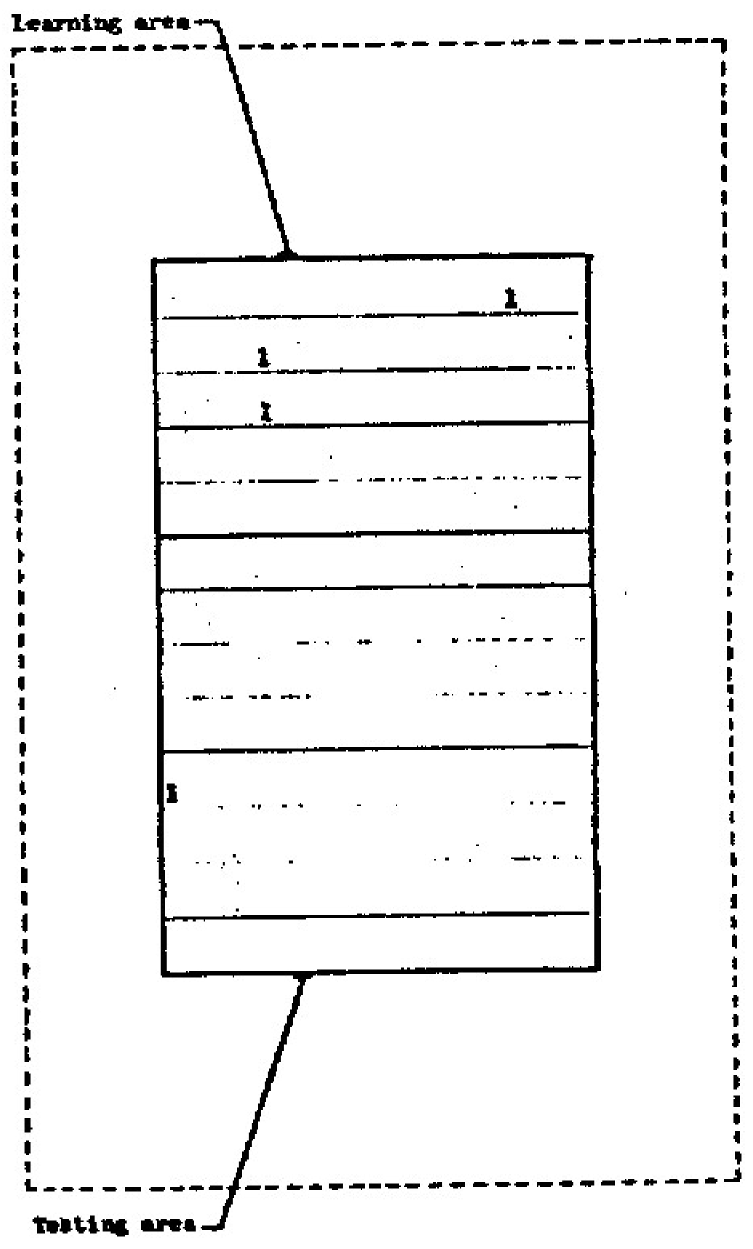
Picture No. 15 (F^1) after applying averaging operator $a_{3,3}$ and thresholding at the level of the main value

(a)



Picture No. 15 (F^1) after applying filters $V_1(F^0/F^1)$ (1st iteration), cl = 0.784, ct = 0.835

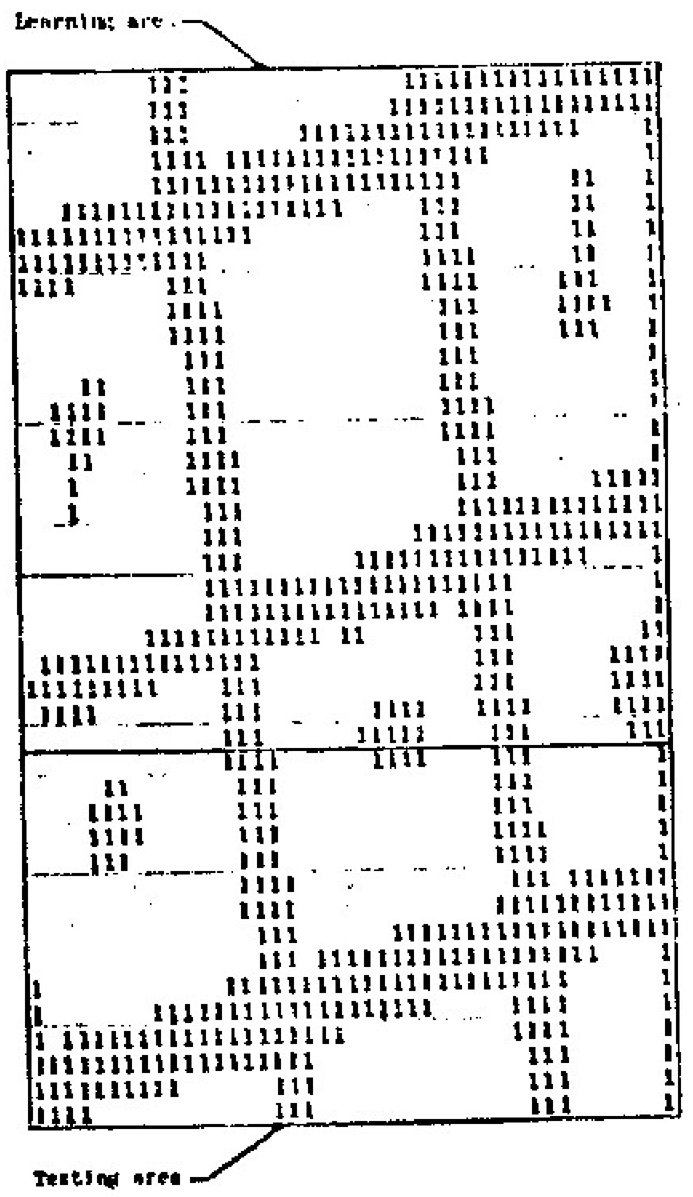
(b)



Picture 15 (F^1) after applying filters $V_{12}(F^0/F^1)$ (2nd iteration), cl = 0.989 ; ct = 0.998

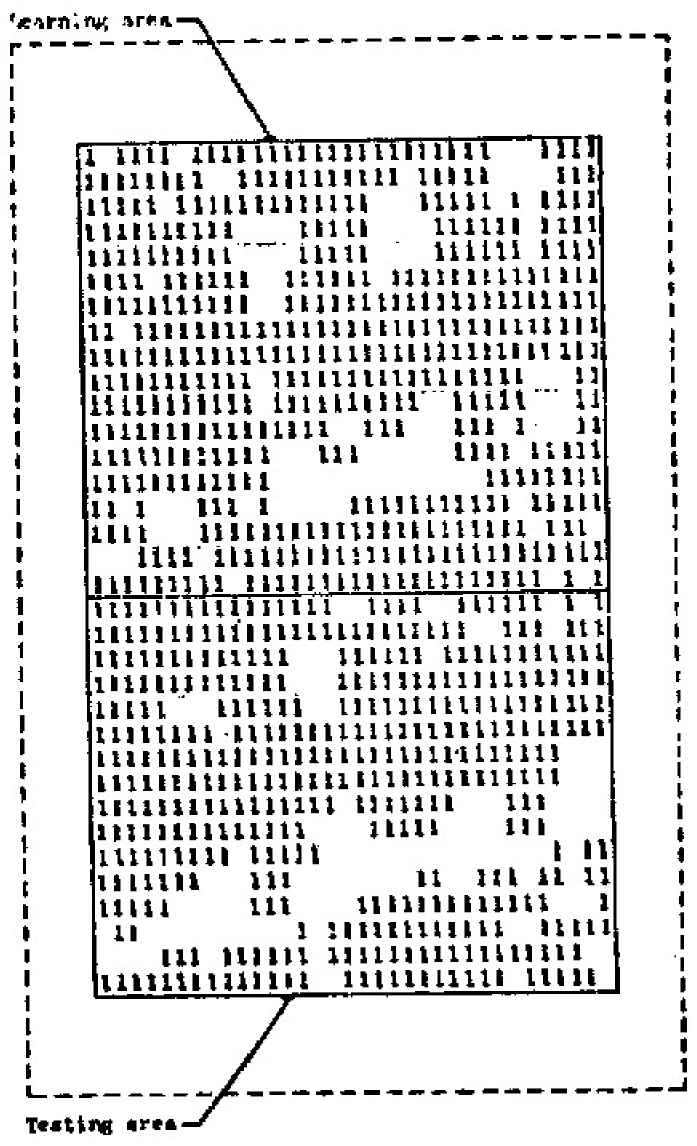
(c)

Figure 3.



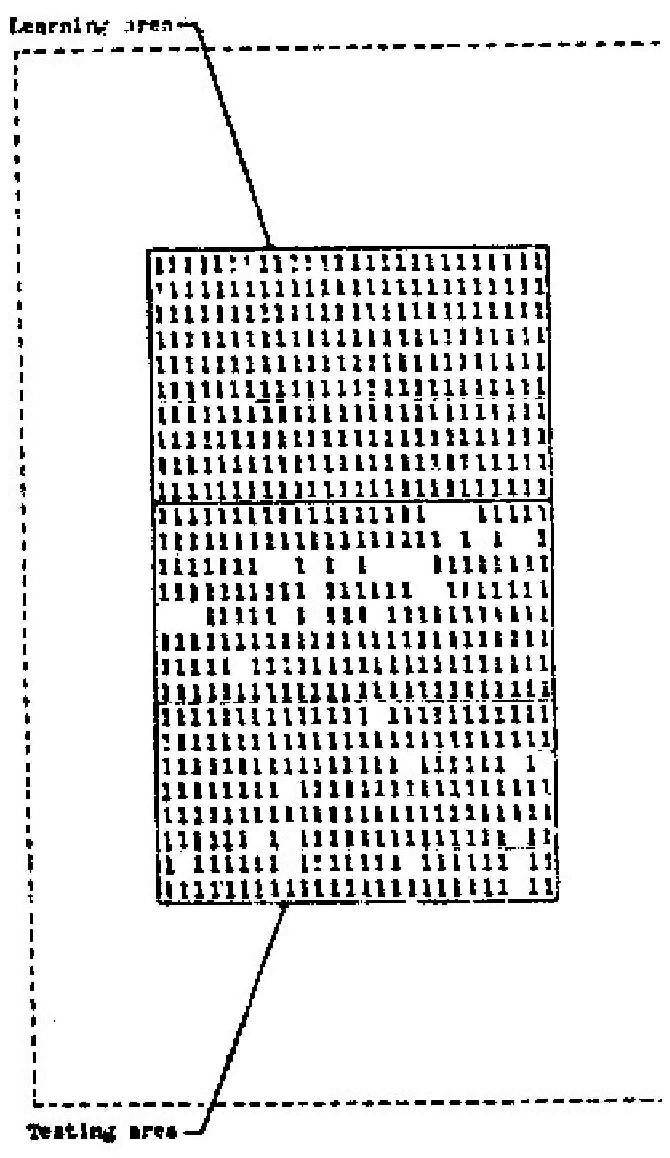
Picture No. 16 (F^1) after applying averaging operator $A_{3,3}$ and thresholding at the level of the mean value

(a)



Picture 16 (F^0) after applying filters $V_1(F^0/F^1)$ (1st iteration), cl = 0.82 , ct = 0.785

(b)



Picture 16 (F^0) after applying filters $V_{12}(F^0/F^1)$ (2nd iteration), cl = 1.0, ct = 0.911

(c)

Job Number	Job Description	Event Extraction Step												
		Pictures	a	ℓ	p	$c(\hat{E}^0)$	$c(\hat{E}^1)$	$c(\hat{E}^\phi)$	Cutting Operator			$c(F^0)$	$c(F^1)$	Execution Time (sec)
									r	lb	ub			
1	1 iteration for F^0/F^1	Pic. 15 '1'-- (grid with noise)	$a_{3,3}$	ℓ_2	p_{12}	169	129	66	2	<0.25	>0.75	81	81	48
2	2 iteration for F^0/F^1	Pic. 16 '0'-- (grid)	$a_{1,1}$	ℓ_2	p_{12}	111	159	2	2	0.25	0.75	111	159	40
3	1 iteration for F^1/F^0		$a_{3,3}$	ℓ_2	p_{12}	225	152	88	3	0.5	0.5	75	92	45
4	2 iteration for F^1/F^0		$a_{1,1}$	ℓ_2	p_{12}	288	296	15	2	0.25	0.75	101	90	41
5	3 iteration for F^1/F^0		$a_{1,1}$	ℓ_2	p_{12}	79	170	3	2	0.25	0.75	26	54	38
6	4 iteration for F^1/F^0		$a_{1,1}$	ℓ_2	p_{12}	17	93	0	2	0.25	0.75	15	37	38

a - averaging operator	r - occurrence threshold
ℓ - grey-level operator	lb - lamba lower bound
p - event-shape operator	ub - lamba upper bound

a.

Job Number	AQVAL/1 Step							
	Event Space and h	Restriction <ms,cs>	Cost-Functional A	VL_1 Formula	# of Terms	# of Selectors	# of Variables	Execution Time (sec)
1	$E(\overbrace{2,2,2,2,2,\ldots 2}^{12\text{ times}})$ 2	<1,1>	c-list:(2,3,5) t-list:(0.1,0.1,0.1)	$V_1(F^0/F^1)$	22	121	12	132
2	"	<1,1>	"	$V_2(F^0/F^1)$	12	68	12	126
3	"	<1,1>	c-list:(1,2,3) t-list:(0.1,0.1,0.1)	$V_1(F^1/F^0)$	23	117	12	138
4	"	<1,1>	c-list:(2,3,5) t-list:(0.1,0.1,0.1)	$V_2(F^1/F^0)$	8	32	12	67
5	"	<1,1>	"	$V_3(F^1/F^0)$	2	4	4	9
6	"	<1,1>	"	$V_4(F^1/F^0)$	2	4	4	6

b.

Job Number	Filtering and Testing Step (application of the operator: a•ℓ•p•v)					
	Operator v	Picture 15		Picture 16		Execution Time (sec)
		Correctness Ratio for Learning Area (cl)	Correctness Ratio Testing Area (ct)	Correctness Ratio for Learning Area (cl)	Correctness Ratio for Testing Area (ct)	
1	$V_I(F^0/F^1)$	0.784	0.835	0.824	0.735	42
2	$V_{II}(F^0/F^1)$	0.989	0.998	1.0	0.911	33
3	$V_I(F^1/F^0)$	0.833	0.870	0.918	0.735	40
4	$V_{II}(F^1/F^0)$	0.958	0.974'	0.946	0.921	31
5	$V_{III}(F^1/F^0)$	1.0	0.997	1.0	1.0	25
6	$V_{IV}(F^1/F^0)$	1.0	1.0	1.0	1.0	26

c.

Summary of the experiment on texture discrimination

Table 4.