78-3

SELECTION OF MOST REPRESENTATIVE TRAINING EXAMPLES AND INCREMENTAL GENERATION OF VL_1 HYPOTHESES: the underlying methodology and the description of programs ESEL and AQ11

by

R. S. Michalski and J. B. Larson

May 1978

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

This work was supported in part by National Science Foundation Grant NSF MCS 76-22940 and in part by a Senior Visiting Fellowship from British Science Research Council.

ABSTRACT

The paper describes the underlying theoretical framework and operational details of two programs, ESEL and AQ11, for computer induction within the framework of the variable-valued logic system VL_1 (i.e., a statement calculus which involves variables with an arbitrary number of discrete values [Michalski 1974]):

ESEL - A supporting program for selecting 'most representative' learning and/or testing VL_1 events from a large data base of events. The program provides the input to the program AQ11:

AQ11 - A program for incremental generation of VL_1 hypotheses, which are generalized and optimized descriptions of input event sets. The program also provides a facility for evaluating the performance of these inferred hypotheses on testing events.

Given a large set of examples describing certain objects or situations, program ESEL selects from them a small subset of the most representative ones. The examples have to be in the form of VL_1 events, i.e., in the form sequences of values of certain discrete variables (or descriptors). In selecting the events, the program distinguishes among three types of descriptors: nominal descriptors, whose value set is an unordered set, linear descriptors, whose value set is a linearly ordered set, and structured descriptors, whose value set is a tree-ordered set.

Events selected by ESEL are input to program AQ11, which generates VL_1 hypotheses describing the events. The program can work incrementally, i.e., given a working hypothesis (a set of rules) obtained at some stage, and a set of events, the program can modify the hypothesis to make it consistent with the events.

Program AQ11 also has the facility to test the performance of a given hypothesis on a set of testing events, and to compute an extended confusion matrix.

1. SELECTION OF THE MOST REPRESENTATIVE TRAINING EXAMPLES: Program ESEL

1.1 Basic Concepts and Notation

The purpose of program ESEL is to select a subset of most representative events from a large number of VL_1 events (see definition below). The need for using this program arises when a given training set of events is very large (say, a few hundred or more events) and AQVAL/1 inductive programs (program AQ11, described here, also AQ7 [Larson, Michalski 75], Uniclass-RS [Stepp 76], AQ9 [Cuneo 75], AQPLUS [Forsburg 75], SYM-4 [Jensen 75], YAL [Yalow 77]) could not accept such a large number of events or would run very inefficiently.

The theoretical background for ESEL is given in [Michalski 75]. Here, for completeness, we will summarize it, and then describe the program itself.

Let $\&(d_1, d_2, \ldots, d_n)$ or, briefly, $\&$, denote a set of all n-tuples $(x_1', x_2', \ldots, x_n')$, $x_i' \in D_i$, $i=1,2,\ldots, n$, where D_i are certain finite sets and d_i is the cardinality of D_i. Thus:

$$\&(d_1, \ldots, d_n) = D_1 \times D_2 \times \ldots \times D_n \qquad (1)$$

$\&$ is called the <u>universe</u> of <u>events</u>, and its elements are called <u>events</u>. x_i' and D_i, $i= 1,2,3\ldots$, denote a <u>value</u> and the <u>domain</u> (<u>value</u> <u>set</u>) of the descriptor x_i, respectively. Descriptors* are certain direct (or derived) measurements or characteristics of objects or situations.

* A descriptor, as described here, is equivalent to a variable. (In a more general sense, not considered here, a descriptor can also be an n-ary relation or n-argument function.)

Depending on the nature of a descriptor, its domain may have a different structure, e.g., it can be a linearly ordered set, a partially ordered set, or an unordered set.

Three categories of descriptors are distinguished here:

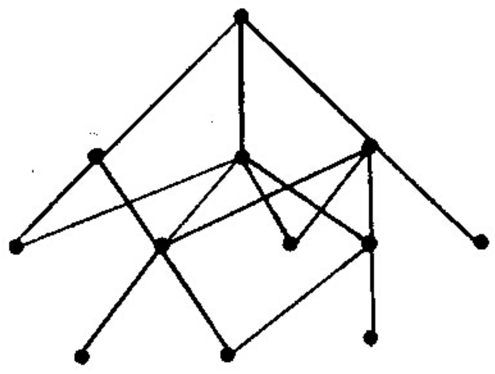
I. Nominal or cartesian descriptors whose domains are sets that have no order.

II. Linear (interval)* descriptors whose domains are any linearly ordered sets. Thus, this category includes ordinal, interval, ratio and absolute variables, as defined in mathematical psychology.

III. Structured descriptors whose domains are partially ordered sets $< S, \geq >$ that are neither linearly ordered nor totally unordered. In this paper we will restrict ourselves to the case of partially ordered sets having the property that for any two elements $a, b \in S$, there exists at least one element c such that

$$a \leq c \text{ and } b \leq c$$

Sets with such structures will be called generalization structures or g-structures. Figure 1 presents a Hasse diagram of a g-structure.



An example of g-structure

Figure 1

* The term used in our previous papers on this subject.

In the diagram, a relation $a \geq b$ is represented by placing node a above node b and linking the nodes by an arc.

> Examples of descriptors: the blood type of a person is a nominal descriptor, the height or weight of a person is a linear descriptor, and the position of the person in an hierarchy of an institution is a structured descriptor.

Suppose, without loss of generality, that we are given two event sets, E1 and E0, where E1, E0 $\subseteq \mathcal{E}$, each associated with a certain decision or action k (k = 1 and 0, respectively). These sets define a set of functions

$$\{f: \mathcal{E} \rightarrow D\} \quad (2)$$

such that

$$\{e \mid f(e) = k\} = Ek,\ k = 1,0 \quad (3)$$

where $e \in \mathcal{E}$ and $D = \{0,1,*\}$; '*' in D means 'no decision'.

A problem of inductive inference is to determine an expression V of a function f, which is most desirable, with respect to some criterion, among all the expressions of all functions (2). Such an expression will usually also assign values 1 or 0 to events not included in Ek; i.e., the expression will be a certain generalization of the sets Ek. Namely, the initial set Ek will be transferred into sets Ek(V) $\supseteq$ Ek, where

$$Ek(V) = \{e \mid V(e) = k\},\ k = 1,0$$

V(e) - the value of the expression V for the event e.

AQVAL/1 programs (Michalski 77) can be used to solve the problem if the expression V is restricted to the class of DVL_1 expressions and the sizes of the sets Ek do not exceed certain limits. When sets Ek are very large (say, a few hundred elements or more), then the computational time of the programs may be too long. The problem arises as to whether sets Ek could not be reduced to more manageable sizes and still provide sufficient information about decision classes from the viewpoint of inductive inference.

If a precise measure of a "degree of representativeness" of each event $e \in Ek$ were available, then an event reduction process could be performed simply by selecting events whose 'degree of representativeness' is above a certain threshold. For example, the frequency of occurrence of an object with the description e in the class k could serve as an estimate of such a measure. This estimate, however, in many practical problems is either not available or is not adequate. Consequently, some other means must be developed for selecting the 'most representative' events.

There can be a number of different methods of solving this problem (see, e.g., [Michalski 75]). Program ESEL implements a method called 'outstanding representatives' (OR).

1.2 An Outline of the OR Method

In this method, the original event set is reduced to a set consisting of events which are most 'distant' from each other. An important feature of this method is that the resulting set will include events which delineate the 'outside' of the events in the original set. For example, if the 'true' but unknown decision class is a circle and its interior and the original event set consists of a number of randomly selected points from this class, then the reduced set will be a set of points lying on or close to the perimeter of the circle and spanning a polygon of approximately equal sides.

This method is, however, very sensitive to events which differ significantly from the rest of the events in the original set. If such events happened to be errors, then these errors would have a strong effect on the result. To circumvent this problem, an additional test could be done, which selects an event only if it has a certain number of 'close' neighbors*. Figure 1 illustrates this method.

*This feature is not implemented.

Let e_1 and e_2 denote two given events:

$$\begin{array}{lll} e_1 = (x'_1,\ x'_2, \ldots,\ x'_{n1}, & x'_{n1+1},\ x'_{n1+2}, \ldots,\ x'_{n2}, & x'_{n2+1},\ x'_{n2+2}, \ldots,\ x'_n) \\ e_2 = (x''_1,\ x''_2, \ldots,\ x''_{n1}, & x''_{n1+1},\ x''_{n1+2}, \ldots,\ x''_{n2}, & x''_{n2+1},\ x''_{n2+2}, \ldots,\ x''_n) \\ \underbrace{\qquad\qquad\qquad\qquad}_{\text{linear variables}} & \underbrace{\qquad\qquad\qquad\qquad}_{\text{structured variables}} & \underbrace{\qquad\qquad\qquad\qquad}_{\text{nominal variables}} \end{array}$$

where x'_i and x''_i denote values of variable x_i in e_1 and e_2, respectively. Assume, without loss of generality, that the first n1 variables in the events above are interval variables, the following n2 variables are structured variables and those remaining are nominal variables†.

First, we will define a measure of the distance $d(x'_i, x''_i)$ between the values of a variable depending on the type of the variable :

- For linear variables:

$$d(x'_i, x''_i) = \frac{|x'_i - x''_i|}{\lambda_i} \quad , \ 1 \leq i \leq n1 \qquad (4)$$

assuming that the domain of each linear variable is represented by the set $\{0, 1, 2, \ldots, \lambda_i\}$, $\lambda_i = d_i - 1$ (d_i – the cardinality of D_i, i.e.of the domain of x_i)

- For structured variables:

$$d(x'_i, x''_i) = \frac{NB}{MNB} \qquad (5)$$

$n1 < i \leq n2$ (see Figure 2)

where NB is the number of branches on the shortest path linking x'_i with x''_i in the Hasse diagram representing the domain of x_i, and MNB is the maximum number of branches on the shortest path linking any two nodes of the diagram.

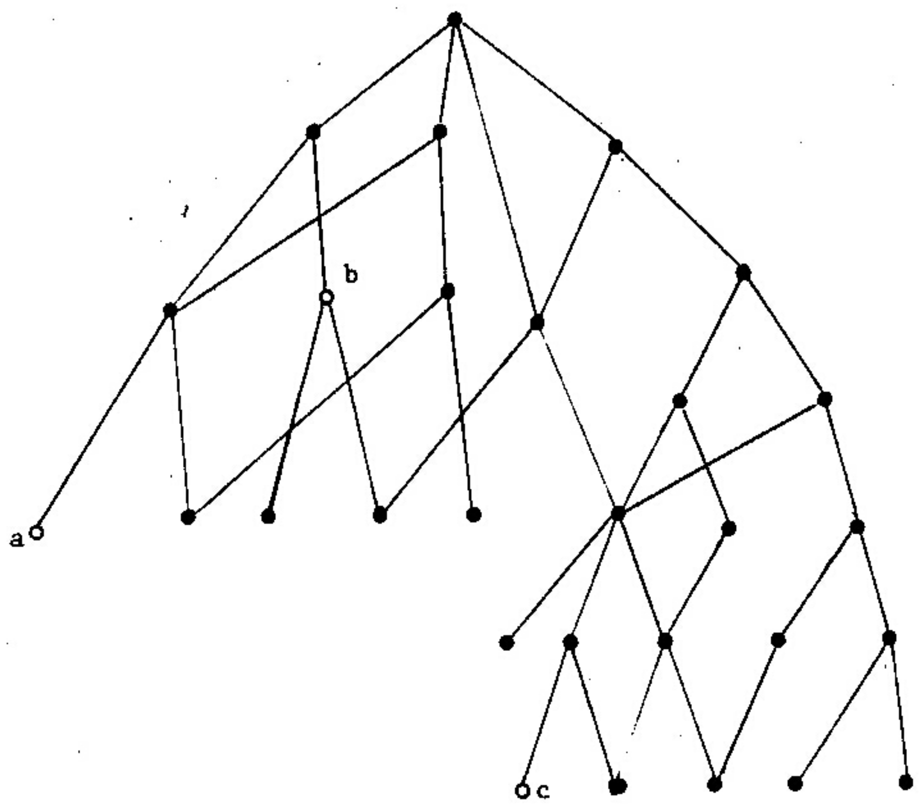
- For nominal variables:

$$d(x', x'') = \begin{cases} 1, & \text{if } x'_i \text{ is not identical to } x''_i \\ 0, & \text{otherwise} \end{cases}$$

$(n2 < i \leq n)$

Two types of distance measures between events are considered:

†It is assumed here that if the domain of a structured variable is not a g-structure, then the variable is treated as a cartesian variable.



NB(a,b) = 3		d(a,b) = 3/9
NB(b,c) = 6	MNB = 9	d(b,c) = 6/9
NB(a,c) = 7		d(a,c) = 7/9

Illustration of a distance between values of a structured variable

Figure 2

(1) Quantized measure:

$$d_q(e_1,e_2) = \sum_{i=1}^{n2} q(d(x_i',x_i''),T_i) + \sum_{i=n2+1}^{n} wd(x_i',x_i'') \qquad (6)$$

where $T_i = (t_{i1}, t_{i2},\ldots, t_{ip})$ is a sequence of thresholds t_{ij} associated with variable x_i, $i = 1, 2, \ldots, n2$

q is a quantization function $q: \{d(x_i', x_i''), T\} \rightarrow \{0, 1, 2,\ldots, p\}$ defined as

$$q(d(x_i', x_i''),T) = \begin{cases} 0, \text{ if } d(x_i', x_i'') \le t_{i1} \\ 1, \text{ if } t_{i1} < d(x_i', x_i'') \le t_{i2} \\ \cdot \\ \cdot \\ \cdot \\ p, \text{ if } t_{ip} < d(x_i', x_i'') \end{cases}$$

w - a 'weight' assigned to nominal variables in relation to non-nominal variables.

(2) Continuous measure

$$d_c(e_1, e_2) = \sum_{i=1}^{n} w_i d(x_i', x_i'') \qquad (7)$$

where w_i is a weight associated with the variable x_i.

The threshold sequence T_i in the quantized measure and weight w_i in the continuous measure represent two different means to control the effect of a single linear or structured variable on the distance between events.

As we can see, control by a threshold sequence avoids a multiplication operation in computing the distance, unlike in control by weight, and thus is computationally simpler than the latter. It requires, however, that the user specifies the value of p and p thresholds for each variable, as opposed to the single number (weight) required in control by weight.

1.3 Algorithm

We will now describe a specific algorithm implementing the OR method. The algorithm is applied in the same way to every set E^k, $k = 1, 2, \ldots$. Let us then assume that E stands for any one of these sets. Either of the distance measures introduced in section 1.2 can be used in the algorithm.

1. For each $e \in E$ determine the distance $d(e, e_0)$, where $e_0 = (0, 0, 0, \ldots, 0)$.
2. Find events e_{min} and e_{max} such that
   $$d(e_{min}, e_0) = \min_{e \in E} d(e, e_0)$$
   $$d(e_{max}, e_0) = \max_{e \in E} d(e, e_0)$$
3. Determine the distance $d(e_{min}, e_{max})$ and divide it into r intervals*, where r is between 0.01 and 0.1 of the size c(E) of the original set E (e.g., if c(E) = 3000 then r is between 30 and 300).
4. Partition E into r subsets, $E_1, E_2, \ldots, E_r$, such that E_i consists of events whose distance $d(e, e_0)$ lies in the ith interval, $i = 1, 2, \ldots, r$:
   $$a_{i-1} < d(e, e_0) \leq a_i$$
   where a_{i-1} and a_i are the endpoints of the ith interval ($a_0 = d(e_{min}, e_0)$ and $a_r = d(e_{max}, e_0)$).

* The intervals do not have to be equal. The desired situation here is to have intervals which will lead to the subsets E_i (determined in step 4) of approximately the same size.

5. From each subset E_i, $i = 1, 2, \ldots, 4$, select a subset E_{is} consisting of s events (where s is such that $r \cdot s$ gives the desired size of the reduced event set). The selection is made in the following way:

   1.) Find e_1 and e_2 in E_i such that

   $$d(e_1, e_2) = \max_{e_a, e_b \in E_i} d(e_a, e_b)^*$$

   2.) Find e_3 such that

   $$d(e_3, e_1) \cdot d(e_3, e_2) = \max_{e \in E_i} (d(e, e_1) \cdot d(e, e_2))$$

   .
   .
   .

   s-1.) Find e_s such that

   $$\prod_{j=1}^{s-1} d(e_s, e_j) = \max_{e \in E_i} \prod_{j=1}^{s-1} d(e, e_j)$$

   where Π denotes the arithmetic multiplication.†

6. The union of the sets E_{is}:

$$E_s = \bigcup_{i=1}^{r} E_{is}$$

gives the reduced event set.

*A more computationally efficient process, though one which might lead to a less desirable result, is to replace step 1 by two steps:
   1a) find e_1 such that

   $$d(e_1, e_0) = \min_{e \in E_i} d(e, e_0)$$

   1b) find e_2 such that

   $$d(e_1, e_2) = \max_{e \in E_i} d(e, e_1).$$

†The reason for using multiplication in steps 2, ..., s-1, is to select events which are at similar distances from each other.

The number of operations required by the algorithm is approximately:

$$N = c(E) + r\,(c(E_i)) + \sum_{j=2}^{s-1} j(t-j)$$

where $c(E)$, $c(E_i)$ is the cardinality of E and E_i, respectively. (E_i are assumed to be all of equal size.) An 'operation' may involve computing the distance between two events, the comparison of two distances, the comparison of the distance with a threshold, etc. In the modified form of the algorithm we have:

$$N' = c(E) + r \sum_{j=1}^{s-1} jc(E_i).$$

For example, if $c(E) = 3000$, $c(E_i) = 100$, $r = 30$, $s = 10$, then $N = 273000$ ($N' = 268000$), and the cardinality of the reduced set would be $c(E_s) = 300$.

1.4 User's Guide for ESEL

INPUT FILES

PARMSX - A file with information about variables.

INST - A file with information about the sizes of event sets which are in the data base and the sizes of representative sets of events.

EVNT - The file containing the data base.

PARMSX FILE

This file contains the number of variables in the event descriptions in the data base, the range of values for each variable, the domain structure for each variable, and the weight which should be given to each variable. The first number in this file must be the number of variables in each event description. The next three specifications are all in the same form: a number, optionally followed by a set of numbers. The first number may be used to set all values of the range, structure, or weight to a single value. If all values in a specification are to be set to one value, then this first number should be this value and there is no following set of values. If a value is to be specified for each variable, the first number should be (0). For example, suppose there are 3 variables with the following situation:

	x_1	x_2	x_3
max. value	3	2	4
structure	interval	nominal	interval
weight	5	6	3

The file PARMSX would look like this:

$$3 \quad \underbrace{0\ \ 3\ 2\ 4}_{\text{max value}} \quad \underbrace{0\ \ 3\ 0\ 3}_{\text{structure}} \quad \underbrace{0\ \ 5\ 6\ 3}_{\text{weights}}$$

The first 3 indicates 3 variables, the first 0 indicates that the range will be specified for each variable (another value than 0 would indicate that all variable ranges will be of that value). The second 0 indicates that the structure of all variables will be individually specified. An interval structure is specified by the number 3, any other number gives a nominal structure. The final 0 indicates that weights will be specified independently for each variable.

Here is another example: suppose there are 4 variables in the data base with the following characteristics:

	x_1	x_2	x_3	x_4
max. value	3	3	3	3
structure	interval	interval	interval	interval
weight	6	5	1	1

Then the PARMSX file would look like this:

```
4   3   3   0  6 5 1 1
```

The 4 indicates that there are 4 variables describing objects in the data base. The next two 3's indicate that all ranges are from 0 to 3 and that all variables are of interval structure. The 0 indicates that weights will be specified independently.

INST FILE

This file contains information about the number of events in each class in the data base and the number of events from each class the program is to select. Each class is specified by two lines in this file. The first line specifies the number of events in the data base which correspond to the class, the second line specifies the number of partitions and the total number of events which are to be selected from the class. A class with 0 events, 0 partitions, and 0 selected events terminates the file.

For example, a data base with 100 events, the first 50 of which are to be in the first class, the next 20 in the second class and the last 30 in the third may be specified as follows:

```
INST:
        blank - the first line must be blank
        50    - the first 50 events are in the first class
        1 5   - using 1 partition, select 5 events
        20    - the second class has 20 events
        1 6   - using 1 partition, select 6 events
        30    - the last class has 30 events
        3 20  - using 3 partitions, select 20 events
        0
        0 0   - the last class
```

EVNT FILE

This file contains the actual data base. Events are stored as lists of integers. Irrelevant values are stored as -1. The first line of this file must be blank.

For example, a situation with 5 events and 3 variables:

EVNT FILE

```
blank
0  -1  3
2   2  3
3   3  1
3   1  1
0   1  0
```

OUTPUT FILES

OFILEX - A file with the selected events.

TOPT - A file with the remaining events which were not selected.

Each file is in the same form as the input file EVNT except that a class number is appended to the beginning of each event. This output format is compatible with the VL_1 mode of the INDUCE-1 (Larson, Michalski 77, Larson 77 a,b) program and the program AQ11.

2. INCREMENTAL GENERATION AND TESTING OF VL_1 HYPOTHESES: Program AQ11

2.1 Introduction

There are many situations when one starts with certain initial hypotheses about given data and then, in the process of experimenting with these hypotheses, has to modify them in order to preserve consistency with new acquired facts. Such situations arise, e.g., in rule-based expert systems, where in the course of a system's performance some rules are discovered to be incorrect or incomplete and have to be modified.

A process of generating hypotheses (or descriptions) in steps, where each step starts with certain working hypotheses and a set of (new) data and ends with appropriately modified hypotheses, is called an _incremental_ (or _multi-step_) _generation of hypotheses_.

The purpose of program AQ11 is to implement such an incremental generation of hypotheses in the framework of the variable-valued logic system VL_1 (Michalski 74). Although from the viewpoint of the complexity of real scientific research this framework is extremely restricted; nevertheless, it is still sufficiently rich to provide an interesting research subject and, also, to obtain solutions which may have practical applications.

Hypotheses are expressed here as (constant-free) disjunctive normal VL_1 expressions (DVL_1 expressions*). A DVL_1 expression is a disjunction of terms, where a term is a logical product of selectors. A selector is a statement in the form:

$$[x \# R]$$

where x is a unary descriptor (variable)

denotes any of the relational operators $= \neq \geq \leq$

R is a list of constants which are elements of the domain of x (R is called the _reference_ of the selector)

*In the general case, DVL_1 expressions involve constants and are multiple-valued logic expressions [Michalski 74]. Here, for simplicity, we will assume initially, that they are just binary (i.e., either satisfied or not satisfied), and have no constants.

When a DVL_1 expression is evaluated for a given event, selectors are interpreted as conditions (or questions). A selector is satisfied if the value of t variable in the event satisfies the condition, otherwise, it is not satisfied. Some examples of selectors and their interpretation as conditions follows:

$[x_i = 1]$	is x_i equal to 1?
$[x_i = 1,3]$	is x_i equal to 1 or 3?
$[x_i = 1..3]$	is x_i between 1 and 3, inclusively?

An example of a term:

$[x_1 = 3][x_3 = 2,4,5][x_5 = 0]$

The above term is satisfied if x_1 equals 2, x_3 has value 2, 4, or 5 and x_5 has value 0.

An example of DVL_1 formula:

$T_1 \ \vee \ T_2 \ \vee \ T_3$

where T_1, T_2, T_3 are terms. The formula is satisfied if term T_1 or T_2 or T_3 is satisfied.

A DVL_1 formula is interpreted as a description of a set of events, namely events which satisfy it.

2.2. Description of Methodology

Suppose there is given a set of hypotheses (DVL_1 descriptions), $V = \{V_i\}$, $i=1,\ldots,m$, and a family of event sets ('facts'), $F=\{F_i\}$, which these hypotheses are supposed to describe. Suppose that for any i, V_i describes correctly only a part of the events from F_i.

The problem is to produce a new set of hypotheses, $V^1 = \{V_i^1\}$, where each V_i^1 describes all events from set F_i, and does not describe events from any other event set F_j, $j \neq i$.

The following solution to this problem is based on the multiple application of a computer program implementing an efficient algorithm [Michalski 71] for determining a cover, $C(E_1/E_o)$, of an event set E_1 against the event set E_o.

Such a cover can be interpreted as a DVL_1 expression, which is satisfied by every event in E_1 and not satisfied by any event in E_o (or in $E_o \setminus E_1$, if E_o and E_1 intersect).

The solution consists of 3 major steps:

Step 1. The first step isolates those facts which are not consistent with the given hypotheses. For each hypothesis, two sets are created:

F^+ - a set of events which should be covered by the hypothesis, but are not

F^- - a set of events which are covered by the hypothesis, but should not be covered.

(An event is said to be covered by a hypothesis if the event satisfies the VL_1 formula which represents the hypothesis.)

Specifically, this step determines, for each i, i=1,2,...,m, the sets*:

$$F_i^+ = F_i \setminus \hat{V}_i \tag{8}$$

$$F_{ij}^- = \hat{V}_i \cap F_j,\ j=1,2,\ldots,m;\ j \neq i \tag{9}$$

(see Figure 3).

Thus, F_i^+ denotes events which should be covered by V_i but are not, and F_{ij}^- denotes 'exception' events, i.e., events in F_i, j≠i, which are covered by V_i, but should not be covered.

Step 2. This step determines, for each i, a generalized formula V_i^- describing all exception events (the union of sets F_{ij}^-, j=1,2,...,m, j≠i). This is done by generating, for given i and each j, a cover of F_{ij}^- against the events in the sets $\hat{V}_i \cup F_i^+$, i=1,2,...,m:

$$V_{ij}^- = C(F_{ij}^- / \bigcup_{i=1}^{m} V_i \cup F_i^+) \tag{10}$$

and then taking the logical union of V_{ij}^-:

$$V_i^- = \bigvee_{\substack{j=1 \\ j \neq i}}^{m} V_{ij}^- \tag{11}$$

*$\hat{V}_i$ denotes the set of events covered by formula V_i.

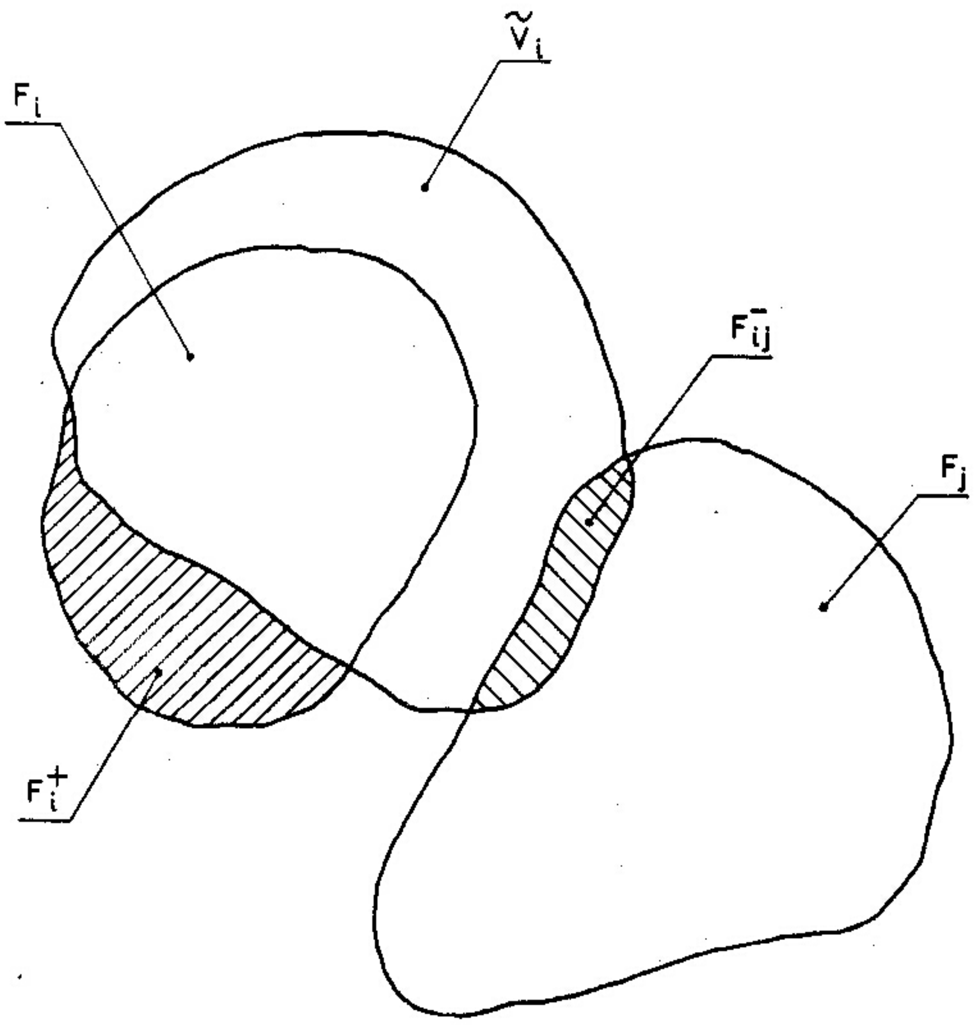


Illustration of sets F_i^+ and F_{ij}^-

Figure 3

The reason for this step is that it is computationally more efficient to use formulas V_i^- than the union of E_{ij}^-, $j=1,2,\ldots,m$; $j\neq i$.

Step 3. New 'correct' hypotheses could be obtained now by 'subtracting' from each V_i the formula V_i^- and 'adding' to it the set F_i^+. To do this directly, however, is difficult. Again, an advantage is taken of the available computer program for generating covers $C(E_1/E_o)$.
Namely, the new hypotheses, V_i^1, $i=1,2,\ldots,m$, are determined as covers:

$$V_i^1 = C(F_i / \bigcup_{\substack{k=1\\k\neq i}}^{m} [(\hat{V}_k \setminus \hat{V}_k^-) \cup F_k]) \qquad (12)$$

(The point is that directly simplifying a union of terms is difficult; but 'substracting' a term from a term or generating a cover of an event set against a DVL_1 formula is easier).

Step 4. This step determines the final representation of hypotheses V_i^1. The V_i^1 are DVL_1 expressions which are unions of terms. Some terms in a V_i^1 may represent (cover) only a few events in F_i. Such 'low weight' terms are replaced by the events (facts) themselves (since an event takes less memory than a term). In program AQ11, parameter PUNY specifies the minimum percent of events which a term has to cover to be a 'high weight' term.
For example, if PUNY = 0.02, and a set F_i has 100 events, then all terms which cover 3 or more events ($3 > 0.02 \times 100$) are 'high weight' terms. Terms which cover 1 or 2 events are replaced with those events.

2.3 An alternative way of handling exception events

In the procedure above, the exception events were represented by terms in V_i^-. If the number of exception events is small, it can be easier to handle the events without turning them into expressions V_i^-. The 'substraction' (denoted by $\setminus$) of an event e from a term T (in a given formula) is done by logically multiplying the term by the negation of the event:

$$T \searrow e = T \wedge \bar{e} \qquad (13)$$

In order to use this way of handling exception events, in program AQ11 the parameter STGY should be set to value 2 (STGY=2).

The result of operation (13) can produce several terms. Anyone of them is sufficient to be used in the new hypothesis. In program AQ11, there is a parameter #EX which specifies how many such terms a user wants to store for representing a hypothesis. If the number of generated terms is larger than #EX, the program selects #EX 'best' terms according to the criteria list.

2.4 Additional Features

There may exist certain restrictions on the event space which must hold in the resulting formulas. A restriction may be of the form

$$[x_3=2] \rightarrow [x_1=NA] \qquad (NA = \text{not applicable})$$

which is read "if x_3 has the value 2 then the variable x_1 is not applicable." The implementation of these restrictions can be viewed as an extra set of hypotheses V_{n+1} which is included in the set E^o of all covers:

$$C(F_i/E^o \cup \tilde{V}_{n+1})$$

Due to the techniques used in the covering algorithm (namely, the use of parameter 'maxstar', see p. 27), this may not be the best approach since only a few terms in each intermediate quantity are retained. Therefore, the program imposes these restrictions on all facts in the set $F=\{F_i\}$. Using the above restrictions, an event

$$e = (1\ 3\ 2)$$

is replaced with

$$e = (NA\ 3\ 2)$$

2.5 Testing Procedure

By applying the above described part of AQ11 program one can determine DVL_1 descriptions (hypotheses) of classes of objects from examples of objects representing individual classes. An obvious problem arises of testing the validity of the derived descriptions. This is done by applying the descriptions to new examples of objects with known class membership. The results of such testing are usually represented in a form of a confusion matrix. This matrix specifies for each class (a row in the matrix), the numbers of testing objects of this class, which were assigned by the descriptions to individual classes (corresponding to columns of the matrix).

Below is an example of a confusion matrix involving 2 classes: a class of cancer cells, and a class of non-cancer cells:

Class (Correct Assignment)	Assigned Decision	
	Cancer cells	Non-cancer cells
Cancer cells	28	2
Non-cancer cells	7	23

Entries on the diagonal indicate the correct decisions, entries outside of the diagonal - incorrect decisions. For example the number 7 in the second row indicates that 7 (testing) non-cancer cells were classified incorrectly as cancer cells.

This form of confusion matrix is adequate if an event (object) either satisfies or does not satisfy a formula. In general, however, it is desirable to consider the degree to which a given event e satisfies or matches a formula. Such a degree, called degree of consonance (or degree of match) and denoted DC(e,V), is computed according to an evaluation scheme. An evaluation scheme consists of definitions for computing:

(1) DC(S,e) - a degree of consonance between a selector and an event (briefly, degree of consonance of a selector),

(2) DC(T,e) - a degree of consonance of a term (a product of selectors),

(3) DC(V,e) - a degree of consonance of a DVL_1 formula (union of terms),

$DC(\{V_i\}, e)$ - a degree of consonance of a set of formulas (describing the same class).

Many different evaluations schemes can be applied for evaluating DVL_1 formulas. Methods developed in many-valued logic (e.g., Recher 69) and fuzzy reasoning (e.g., Zadeh 74, Gaines 76) are applicable here. We will describe the evaluation scheme currently implemented in program AQ11, and give suggestions for other evaluation schemes.

(1) *Definition of degree of consonance of a selector.*

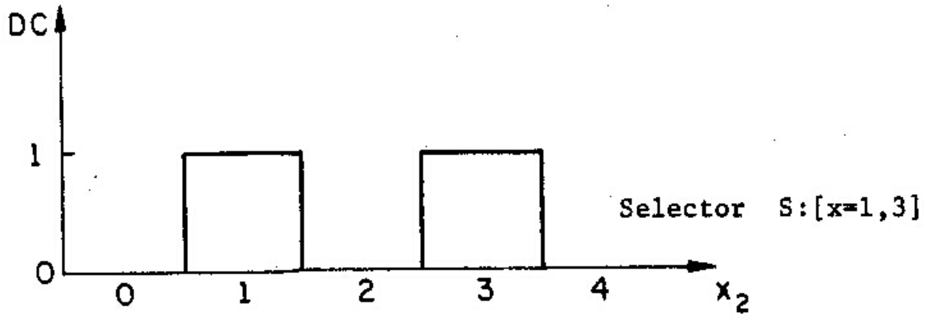
The basic definition of the degree of consonance, DC(S,e), of a selector comes from the evaluation rules in VL_1 [Michalski 74]. Assuming that the output domain of the formulas $D = \{0,1\}$ we have:

$$DC(S,e) = \begin{cases} 1, & \text{if the value of appropriate variable in e satisfies the selector S} \\ 0, & \text{if it does not satisfy S} \\ * & \text{the value is unspecified} \end{cases}$$

For example, suppose event $e = (x_1, x_2, x_3) = (3, 1, 1)$, and selector S is $[x_2=1,3]$. We have $D(e,S) = 1$, because value of x_2 in e is a member of the reference of the selector (i.e., 1 is member of {1,3}). (Fig. 4).

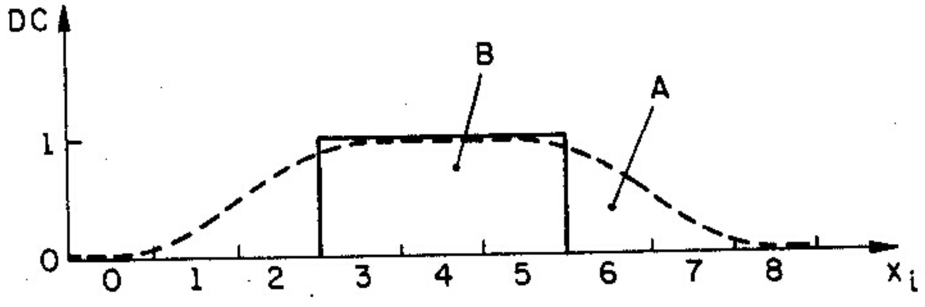
Alternative evaluation schemes can take into consideration the structure of the domainof the variable in the selector.If a variable is linear, it seems that the above definition of DC(e,S) is too rigid. For example, if a linear variable x_i=13 and S: $[x_1=14..18]$, the selector is evaluated to 0, while it seems desirable to evaluate it to some value greater than 0 (since 13 is so 'close' to 14). This means that one could accept a 'bell-shaped' function for evaluating interval selectors (Fig. 5).

The concept of 'trimming' a term can be also useful here. In an untrimmed (extended) term, references of selectors (sets of values) are as large as possible without leading to a contradiction, i.e., intersectionwith formulas of different classes. In the trimmed term, references are as small as possible, providing that the term still covers the same learning events as the extended term and preserves the type



A graphical illustration for evaluation selector $[x_2=1,3]$

Figure 4



A bell-shaped (A) versus step-shaped (B) function for evaluating a linear selector $[x_i=3..5]$

Figure 5

of selectors, e.g., if the reference of a linear selector is $a..b$, then in the trimmed selector it will be an interval $a_1..b_1$, $a \le a_1$, $b_1 \le b$.

An evaluation function can assign DC = 1 when a variable has value within the 'trimmed' reference, assign DC = 0, when the variable has value outside of the extended reference, and assign DC = B, $0 < B < 1$, otherwise. (Fig. 6 a and b).

(2) *Definition of the degree of consonance of a term.*

In the definition of VL_1, the degree of consonance of a term was defined as minimum of values of selectors in the term. In AQ11, the degree of consonance of a term is computed as the ratio of the number of selectors satisfied in the term to the total number of selectors in the term.

If all selectors in a term are satisfied, then both definitions give the same value. If this is not the case, the latter definition differentiates between the terms with different numbers of selectors satisfied, while the former does not, (which is a desirable feature).

As an alternative, one could use here also a probabilistic logic evaluation, which evaluates a term into the arithmetic product of DC-s of selectors.

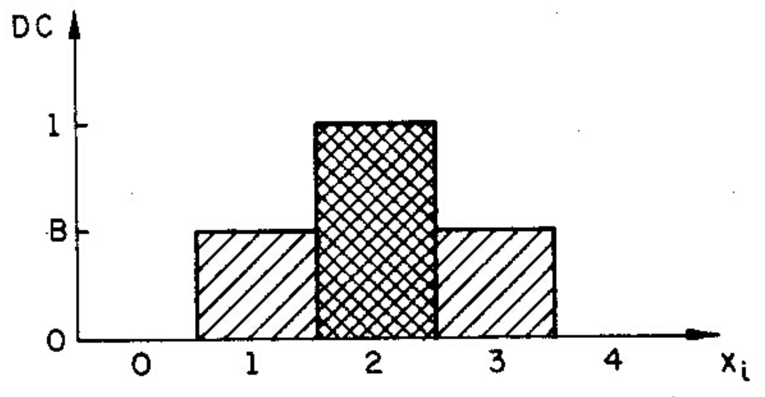
(3) *Definition of the degree of consonance of a formula.*

The degree of consonance of a formula V and an event e, DC(V,e), is defined as the maximum of degrees of consonance $DC(T_i,e)$, of terms T_i, in the formula (i.e., as defined in VL_1), i.e.:

$$DC(V,e) = \max_{T_i \in V} \{DC(T_i,e)\}$$

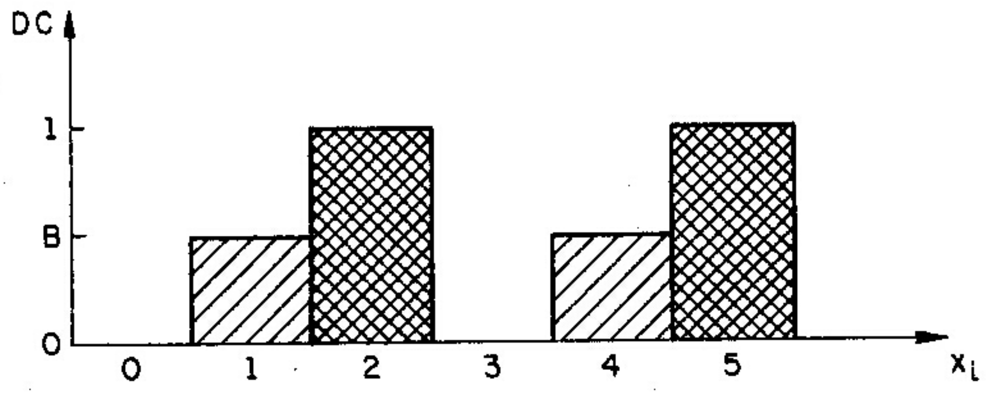
(4) *Definition of the degree of consonance of a set of formulas of the same class.*

It is useful sometimes to generate more than one formula describing a given class. The reason is that having more than one formula per class may



Generalized selector S: $[x_i=1:3]$ Trimmed selector S': $[x_i=2]$

a. Evaluation function for an interval selector using the concept of 'trimming'



Generalized selector
S: $[x_i=1,2,4,5]$

Trimmed selector
S: $[x_i=2,5]$

b. Evaluation function for a nominal selector using the concept of 'trimming'

Figure 6

improve the reliability of decision making. We have accepted in AQ11 the definition that the degree of consonance $DC(\{V_i\},e)$, of a set of formulas, $\{V_i\}$, as the <u>average</u> of the DC-s of the formulas in the set:

$$DC(\{V_i\},e) = \underset{V_i}{AVG}\ \{DC(V_i,e)\}$$

Given a set of formulas of different classes and an event, the DC is computed between the formula (or a set of formulas) of each class and the event. The classification decisions are ordered according to the value of DC. All decisions with value DC within the distance TAU (see parameter TAU in section 2.5) from the maximum value of DC, are rank 1 decisions (i.e., each of these decisions are treated as equally justified). Then the next 'best' decision which is not of rank 1 is selected, and all decisions with DC within TAU distance from DC of the selected decision are assigned rank 2. The process repeats IRK times (see input parameter IRK in section 2.5).

For each testing event, values of DC of ranked decisions are printed by the program AQ11 as rows in an 'extended' confusion matrix (see Fig. 7 for an example). In the matrix, a decision of rank 1 which is correct is underlined, and the number of rank 1 decisions for the given event is printed in the 'TIES' column. If an event has some unspecified values, it may still be possible to compute DC for certain classes, and for certain classes DC would depend on the value of unspecified variables. If a decision of rank 1 is correct and would remain so no matter what values unspecified variables take, then the decision is treated as a correct decision. In any other case, the decision is excluded from computing total correctness statistics, and the column 'UNSP' corresponding to the given event has entry * .

The performance statistics for each group of events of one class are printed in the last 2 rows of the group of rows associated with these events. The rows contain the number and percentage, respectively, of events classified to each class. The percentage of correct decisions is braced by vertical bars. The matrix also contains a column #DEC/event specifying an 'indecision ratio',which is a ratio of all decisions of rank 1 to the total number of events in the group (excluding rows with USP=*). Figure 7 gives an example of an extended confusion matrix. The matrix was computed for 3 classes D1, D2 and D3 described by formulas:

D1: [x1=2][x3=1][x4=1]

D1: [x1=2][x2=0]

D3: [x2=1][x4=1]

and for 3 testing events of class D1:

e1: (1011)

e2: (2111)

e3: (*011) (* denotes unspecified value)

The parameters were: TAU=0.1 and IRK=2:

				Assigned Decision		
Correct Assign	#DEC/Event	TIE	UNSP	D1	D2	D3
				$\underline{.68}$	.50	.50
		2		$\underline{1.00}$	.50	1.00
D1			*			0.50
				2	0	1
	1.33			|67%|	0%	33%

The percentages braced by vertical bars indicate the correct decisions.

An example of an extended confusion matrix

Figure 7

The value of #DEC/Event is 1.33 because there were 4 decisions of rank 1 (including new with UNSP=*) and 3 events (4/3=1.33).

Concluding, the program AQ11 permits a user to determine decision formulas in an incremental way and then automatically test them on the testing data. Thus, it is a 'complete' tool for making experiments in inductive learning of DVL_1 descriptions of data.

2.6 Program User's guide

The program requires two sets of input parameters: control parameters (read in PL/1 data format) and data parameters (read in PL/1 list format). All but two control parameters have default values, therefore only two (NV, NCL) must be specified. All data parameters must be specified (although some may be omitted if certain control parameters are set). The following is a list of all parameters which the program currently accepts. Default values (if any) are given in the examples.

2.6.1 Control parameters

- NV (no default value)

  Example: NV = 50

  Possible values: integer in the range 1:50

  This parameter specifies the number of variables which are available to describe each event.

- NCL (no default value)

  Example: NCL = 19

  Possible values: any positive integer

  NCL specifies the number of classes of events (or event sets) which are to be input to the program. The program will then generate a cover of each event set.

- MAXSTAR

  Example: MAXSTAR = 10 (default value)

  Possible values: any positive integer

  MAXSTAR is the maximum number of complexes (terms) which are kept in any intermediate star (see AQ7 documentation for a further description [Larson, Michalski 75]). The procedure for selecting 'best terms' is somewhat different in this program than in AQ7. (When each intermediate star is trimmed, the user-specified cost function is used rather than only the first two criteria).

- NPASS

  Example: NPASS = 1 (default value)

  Possible values: any non-negative integer

  If NPASS = 0, the program will only execute the confusion matrix phase and then terminate (Set TEST = '1'B).

  If NPASS = 1, the program will form a cover of the facts using input formulas (if any), then evaluate the formulas if TEST = '1'B.

  If NPASS > 1, the program will partition the set of facts into sets whose size depends on the PCT parameter (see below) and form hypotheses based on old hypotheses and the partitioned facts. Then, new sets of events will be taken in turn and hypotheses formed based on the entire set of events taken up to the point and the hypotheses from the last pass.

- TEST

  Example: TEST = '0'B (default value)

  Possible values: '1'B or '0'B

  If TEST is '1'B, then a confusion matrix will be computed after each pass. The testing events must be given to the program in the file TESTF.

- RESTRICT

  Example: RESTRICT = '0'B (default value)

  Possible values: '1'B or '0'B

  If RESTRICT is '1'B, then a set of restrictions is accepted by the program (see parameter REST) and applied to all events.

- RTEST

  Example: RTEST = '0'B (default value)

  Possible values: '1'B or '0'B

  If RTEST is '1'B, the restriction will also be applied to testing events.

- TRANS

Example: TRANS = '0'B (default value)

Possible values: '1'B or '0'B

If TRANS is set to '1'B, then the variable names and values are translated into descriptive names in the output. In this case, a file TRAN must be given to the program (see TRAN below).

- PUNY

Example: PUNY = 0.02 (default value)

Possible values: real value in interval [0:1]

All terms which cover less than a percent (PUNY*100) of the events of the corresponding set will be discarded in the next pass (i.e., if a term covers 2 events, PUNY = 1, and there are 23 events in the training set, this term will not be used in the next pass).

- TAU

Example: TAU = .019 (default value)

Possible values: real values in interval [0:1]

This parameter relates to the computation of the confusion matrix. Any two values (degrees of consonance) within TAU of each other are considered to be of the same rank. For example, if .98 is the highest decision value for a testing event, any decision with a value between .96 and .98 would be a rank 1 decision (assuming default value of TAU).

- IRK

Example: IRK = 2 (default value)

Possible values: positive integer less than NCL

This parameter also relates to the computation of the confusion matrix and controls the number of decisions printed out. All degrees of consonance of rank not greater than IRK are printed, others are not printed. If IRK = 1, only rank 1 degrees are printed. One exception to this is that the degree associated with the correct decision is always printed.

- NCRIT

  Example: NCRIT = 2 (default value)

  Possible values: integer in range [1:4]

  NCRIT specifies the number of cost criteria which should be applied when computing the cost of a formula (see CRIT). CRIT(1) through CRIT(NCRIT) will be used, all others will be ignored.

- CRIT(I)

  Example: CRIT(1) = 1 CRIT(2) = 2 (default value)

  Possible values: each CRIT(I) may have values 1, 2, 3, 5 and 9

  CRIT(I) = J specifies that the I-th criterion in order will be the cost function J. There should be NCRIT specifications indicating the cost function which will be used (J) and the order in which they will be applied (I). Available cost functions are the following:

  1. Maximize the number of events covered by the given term, and not covered by previous terms

  2. Minimize number of selectors

  3. Minimize cost of all variables in this term. If this criterion is specified, costs of variables must also be specified (see Z parameter)

  5. Minimize the number of events of E0 covered

  9. Maximize total number of events covered by a term

- #EX

  Example: #EX = 1 (default value)

  Possible values: positive integer

  During some phase of the program, exception terms are formed (description of events which are covered by hypotheses but should not have been). #EX gives the number of redundant exception terms (i.e., the terms which cover the same event).

- TR

Example: TR = '0'B (default value)

Possible values: '1' or '0' B

TR gives a trace of the multi-step process giving the exception terms and the size of the sets F^+ and F^- described in Section 2.1.

- STGY

Example: STGY = 1 (default value)

Possible values: 1 or 2

If STGY has the value 1, then exception terms are formed for events in the sets F^-. If STGY has the value 2, then the previous hypotheses are multiplied by the complement of the exception events of the set F^-.

- INDEP

Example: INDEP = '0'B (default value)

Possible values: '0'B or '1'B

If INDEP is '1'B, then the number of independently covered events are printed for each complex. Otherwise, only the number of new events and the total number of events covered are printed.

- TITLE

Example: TITLE = 0 (default value)

Possible values: non-negative integer

TITLE specifies the number of cards which are in the title. The title cards must follow the semi-colon which terminates the set of control parameters.

- OPT

Example: OPT = '1'B (default value)

Possible values: '1'B or '0'B

If OPT is '1'B, then after each pass a table is printed indicating the numbers of times each cost criterion is evaluated (number of terms for which the cost function is evaluated).

- MODE

  Example: MODE = 'IC' (default value)

  Possible values: 'IC', 'DC', 'VL'

  If MODE = 'IC', then covers are allowed to intersect over 'DON'T CARE'areas of the event space. If MODE = 'IC', the covers are constrained to be disjoint. MODE = 'VL' gives order dependent covers.

- CPXEV

  Example: CPXEV = '1'B (default value)

  Possible values: '1'B or '0'B

  If this parameter is '1'B, then during the testing phase a table is printed which gives the number of times each term was needed to give a correct decision.

- GEN

  Example: GEN = '1'B (default value)

  Possible values: '1'B or '0'B

  If this parameter is '1'B, then only the necessary parts of the reference of each output complex are printed (i.e., a new term is created from the generated term which has the following properties):

  a. The new term covers the same events.

  b. The new term contains the same variables.

  c. The references in the new term are as small as possible.

- ECHO

  Example: ECHO = 'ERZ' (default value)

  Possible values: A string contains any of the characters ZERF

  If the letter appears, the corresponding input data is echoed.

  E = Events
  R = Restrictions
  Z = Variable costs
  F = Input formulas

  The default echos events, restrictions and variable costs if they are in the input.

- TOLERANCE(I)

Example: TOLERANCE(2) = 0.0 (default value)

Possible values: integer or real in [0:1]

TOLERANCE(J) is the tolerance for the J-th criterion specified. If it is an integer, then it is assumed to be an absolute tolerance. Otherwise, it is a relative tolerance calculated by finding TOLERANCE * (MAX-MIN) when MAX or MIN are the maximum and minimum elements in the list of costs to be sorted.

- ORD

Example: ORD = '1'B (default value)

Possible values: '1'B, '0'B

If ORD is '1'B, then the program will reorder events in EO, in decreasing order, with regard to the distance from e_1.

- N-TAU

Example: N-TAU = 0 (default value)

Possible values: integer in [0:8]

This parameter, if not zero, generates a TAU estimation table giving summary information for each class in the evaluation procedure using N-TAU values of TAU beginning at 0 with increments of TAU-INC.

- TAU-INC

Example: TAU-INC = .02 (default value)

Possible values: Real in [0:1]

This is the increment used in the TAU estimation table.

Semi colon (;): This must be entered to terminate the control parameters.

2.6.2 Data parameters

These parameters have the names as used in the program. In the input to the program only their values are specified, in the order given here (See fig. B-1 (a) for an example.)

- TITLEC

  Possible values: The number of lines specified by the TITLE parameter

  These lines are printed at the top of the output.

- NSPEC

  Possible values: An integer in the range [0:NV]

  Number of variables for which a structure is to be specified.

- VTYPE

  Possible values: 'F', 'I'

  The NSPEC variables will be of this type ('F' - nominal variable, 'I' -linear variable).

- TYPE

  Possible values: A list of NSPEC integers in the range [1:NV]

  The list indicates variables of VTYPE.

  Example of NSPEC, VTYPE, TYPE: 3'F' 1 3 5

  There are 3 variables of type 'F' (nominal) namely, variables 1, 3, and 5. The rest will have type 'I'.

- NL

  Possible values: A list of NV positive integers in the range [1:8]

  This parameter gives the number of values which each variable can assume.

  Example: 1 2 4

- NE

Example: 3 1 4

Possible values: A list of NCL integers in the range [0:NEVE]

The parameter specifies the number of events in each event set. The sum should add up to NEVE.

- NF

Example: 3 4 1

Possible values: A list of NCL non-negative integers

This parameter specifies the number of terms of the hypothesis for each event set.

- PCT

Example: .2 .4 1

Possible values: A list of NPASS real values in range [0:1] (except if NPASS = 1, PCT is assumed to be 1)

In this example, 20% of the events will be described first, then an extra 20% of the events will be added and a description formed using previous hypotheses. Finally, the complete set of events is used (see NPASS above).

- REST

Example: (x12 = 1) -> (x14 = *);
(x13 = 2) -> (x1 = *) (x4 = 1).

Possible values: A list of decision rules separated with semi-colons and terminated with a period

This restriction will be applied to all events (i.e., added to current specifications). RESTRICT must be set to specify restrictions. An * in the reference indicates that this variable is not applicable. Restrictions are separated by semi-colons and the list of all restrictions is terminated by a period.

- EVENT

Possible values: NEVE lists of events, NEVE = SUM(NE)

There are two ways in which events can be specified, and the two types of specifications can be mixed.

1. An event can be specified as a list of values, one value for each variable. The values can be:

   a) non-negative integer--indicating value of the variable
   b) -1--variable does not apply
   c) -2--do not know the value

   Example:
```
3  2  0  -1  -2  0
4  1  2   0   0  0
```

2. An event can also be specified by a VL1 formula which is preceeded by a line which says FORMULA. Each formula must be terminated by a semi-colon.

   Example:
```
FORMULA
(x1 = 2) (x3 = 0);
FORMULA
(x3 = 1) (x21 =2);
```

- FORMULA

Possible values: NCL lists of formulas, each having NF complexes

There are two ways to specify a formula:

1. as a FORMULA as in the event specification,

2. as a binary positional bit string in PL/1 List Format.

- Z

Example: Z(1,2) = 9   Z(3,4) = 57;

Possible values: Integer values terminated by semi-colon in PL/1 Data Format

These are costs of the variables which are accepted if CRIT(I) = 3 has been specified for the event set I. If Z value is not specified for some variables, it is assumed to be 1. Z(I,J) = Y means that variable x_J has cost Y for event set I.

2.6.3 Files

- TEST

  This file must be included if the parameter TEST is set to '1'B. The first line of this file contains a list of NCL values indicating the number of test events for each event set. The list of testing events follows. Each event is specified as a list of variable values with coding of -1 and -2 as above.

- TRAN

  This file must be included if TRANS is '1'B. It contains the names of all variables and variable values. Each name will be truncated: variable names to 20 characters, value names to 10 characters. The format is the following: For each variable one specifies:

  variable name, variable value names

  Each name must be in single quotes.

  Example:

```
//TRAN DD *

    'TEMPERATURE'

            'COLD'
            'MODERATE'
            'WARM'

    'HUMIDITY'

             'DAMP'  'DRY'
```

- TESTF

  This is a temporary file which the program uses to store test events from one pass to the next. See JCL set up for specification of this file.

This completes a description of the input specification to the program AQ11. For a user's convenience, appendix A gives a summary of the input specification. Appendix B gives an example of input and output from the program.

2.6.4 Program Output

Most of the output is self-explanatory (see appendix B). The input data is echoed when specified. Then, the formulas for each pass are printed. To the right of each term is a pair of numbers which specify the number of new events covered and the total number of events covered by that term.

After all the formulas for one pass are printed, a confusion matrix is printed for these formulas and given testing data. Information about each pass is printed in turn until all passes are complete.

If two events of different classes are identical, then a message is printed indicating a non-disjoint representation of classes. In such a situation, if a cover C(E1/E0) is being created, then the event of E0 is ignored.

The output from the evaluation part of the program consists of an extended confusion matrix, as described in section 2.5.

Two other tables are printed at the user's option. If CPXEU is set, then a table listing the number of correct decisions for each complex is given. If N-TAU is not zero, then TAU estimation table is printed, giving the indecision ratio and number of correct decisions for each class for N-TAU values of TAU, beginning with 0 in increments of TAU-INC.

3. SUMMARY

We have described here the underlying methodology and computer programs for selecting 'best' learning VL_1 events (program ESEL), and incrementally generating VL_1 hypotheses for given event sets (e.g., selected by program ESEL), and then automatically testing them on the supplied testing events (program AQ11). These two programs constitute a package which can be used for making experiments in induction of descriptions from examples in various applied fields.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation under Grant NSF MCS 76-22940, and in part by a Senior Visiting Fellowship from the Science Research Council in U. K.

The paper was written while one of the authors, R. S. Michalski, was spending his sabbatical leave at the University of Essex in England. This author would like to express here his deepest gratitude to Professor Brian Gaines, the head of Electrical Engineering Department of University of Essex University, for unusual hospitality and help to organize life in the new environment, as well as for the numerous and inspiring discussions.

Thanks are also due to Tom Dietterich for the strenuous job of proofreading of this paper.

REFERENCES

1. Cuneo, R. P. Selected Problems of Minimization of Variable-Valued Logic Formulas. Report No. 726, Department of Computer Science, University of Illinois, Urbana, Illinois, 1975.

2. Forsburg, A. S. A user's guide for AQPLUS, on internal report, Department of Computer Science, University of Illinois, Urbana 1975.

3. Gaines, B. R. Foundations of Fuzzy Reasoning, International Journal of Man-Machine Studies, No. 8, 1976.

4. Jensen, G. M. Determination of Symmetric VL_1 Formulas: Algorithm and Program SYM4. Report No. 774, Department of Computer Science, University of Illinois, Urbana, Illinois, 1975.

5. Larson, J. Inductive Inference in the Variable Valued Predicated Logic System VL_{21}: Methodology and Computer Implementation, Report No. 869, Department of Computer Science, University of Illinois, Urbana, Illinois, 1977.

6. Larson, J. Induce 1: An Interactive Inductive Inference Program in VL_{21} Logic System. Report No. 876, Department of Computer Science, University of Illinois, Urbana, Illinois, 1977.

7. Larson, J., Michalski, R. S. Inductive Inference of VL_1 Rules. Workshop on Pattern Directed Inference Systems, Honolulu, Hawaii, May 1977.

8. Larson, J., Michalski, R. S. AQVAL/1 (AQ7) User's Guide and Program Description. Report No. 731, Department of Computer Science, University of Illinois, Urbana, Illinois, 1975.

9. Michalski, R. S. TOWARD COMPUTER-AIDED INDUCTION: A Brief Review of Currently Implemented AQVAL Programs, Report No. 874, Department of Computer Science, University of Illinois, Urbana, Illinois, May 1977.

10. Michalski, R. S. On the Selection of Representative Samples from Large Relational Tables for Inductive Inference, Department of Information Engineering, University of Illinois at Chicago Circle, July 1975.

11. Michalski, R. S. VL_1: Variable-Valued Logic System. 1974 International Symposium on Multiple-Valued Logic, West Virginia University, Morgantown, West Virginia, May 1974.

12. Michalski, R. S. A Geometrical Model for the Synthesis of Interval Covers. Report No. 461, Department of Computer Science, University of Illinois, Urbana, Illinois, 1971

13. Rescher, M. Many-Valued Logic. McGraw-Hill, New York, 1969.

14. Stepp, R. Uniclass Cover Synthesis: Methodology and a Computer Program Description, Report No. , Department of Computer Science, University of Illinois, Urbana, Illinois, .

15. Zadeh, L. A. Fuzzy Logic and its Application to Approximate Reasoning, Proceedings IFIP Congress 1974, Vol. 3, North-Holland, 1974.

APPENDIX A

AQ11 Input Specifications

1. ID parameters

Allow 150 to 180 K Bytes of storage for large problems. A very small problem may be run in 120 K. Very few IOREQ's are used by the program; 500K ismore than enough. Time is the main variable which must be adjusted. Using the following parameters, an estimate of the time required for a large job can be given.

MAXSTAR = 1	NPASS = 3	NV = 35
NCL = 19	NEVE(training) = 307	No evaluation
Time: 1 min.	Region: 174K	

Changing MAXSTAR to 7 and requesting evaluation using 388 events, the time increased to 3 minutes for the training phase and 1 minute and 30 seconds for evaluation.

2. JCL

The following JCL is recommended:

```
//   EXEC PGM=ITCN3,REGION=180K,PARM='ISA(N),REPORT'
//STEPLIB DD DSN=USER.P2123.ITCN3,DISP=SHR
//SYSPRINT DD SYSOUT=A
//PLIDUMP DD SYSOUT=A
//SAVEF DD DSN=&&TEMP,UNIT=DISK,SPACE=(TRK,(10,1))
//FT06F001 DD SYSOUT=A
//SYSIN DD *
        input parameters and data

//TEST DD *
       test data (if evaluation requested)

//TRAN DD *
       translation data (if TRANS is set)
```

ISA(N): N should be the region requested minus 125,
e.g. 51K

3. Control parameters

Parameter	Default	Description
NV	---	Number of variables
NEVE	---	Total number of training events
NCL	---	Number of classes
MODE	'IC'	Mode of operation
MAXSTAR	10	Maximum star size
ECHO	'ERZ'	Echo input
NCRIT	2	Number of criteria
CRIT(1)	1	Criterion 1
CRIT(2)	2	Criterion 2
CRIT(3)	5	Criterion 3
CRIT(4)	9	Criterion 4
TITLE	0	Number of lines in title
RESTRICT	'0'B	Accept restrictions
SAVE	'0'B	Save formulas in a file SAVEF
GEN	'1'B	Trim complexes for output and evaluation
PUNY	.02	The minimum percent of events which have to be covered by a term
TR	'0'B	Trace multi-step procedure
NPASS	1	Number of steps
STGY	1	Way in which events of F^- sets are handled
#EX	1	Numbers of redundant exception complexes
OPT	'1'B	Print statistics about number of times each cost function is evaluated
TRANS	'0'B	Translate output using TRAN file
TEST	'0'B	Evaluate formulas
RTEST	'0'B	Apply restrictions to test events
TAU	.019	Equivalent threshold for rank 1 decisions
IRK	2	Number of ranked decisions which are printed
CPXEV	'1'B	Print statistics about satisfied complexes during evaluation
NGE	200	Initial storage for complexes
INDEP	'0'B	Prints independent events if set
TOLERANCE(I)	0	Tolerance for I^{th} specified test function
N-TAU	0	Number of columns in 'tau' estimation table
TAU-INC	.02	Increment in tau estimation table
ORD	'1'B	Reorder the events in E0, in decreasing order, with regard to the distance from e_1.
Semi-colon (;)		Terminate control parameters

4. Data parameters

Parameter	Description
TITLE	Lines of title (if any)
NSPEC	Number of variables for which type TYPE is specified
TYPE	Type of these variables

VSPEC	Indicies of variables of type TYPE
PCT	If NPASS > 1, the percent of events to use in learning phase for each pass
NL	Number of values for each variable
NE	Number of events in each set
NF	Number of formulas in each set
RESTRICTIONS	If RESTRICT is set. Each pair of rules must be separated with a semi-colon; the entire list is terminated with a period.
EVENTS	Lists of events in either of two forms
FORMULAS	Lists of formulas as in either of two forms
Z	If any CRIT(I) = 3, costs of variabl terminated with semi-colon

5. Files

<u>File</u>	<u>Description</u>
TRAN	TRANS is '1'B, the file names of classes, variables and variable values. Each name must be in quotes
TEST	TEST is '1'B, the file of test events
SAVEF	SAVE is '1'B, the output file of formulas in bit positioned form (list format)

APPENDIX B

An Example of an Input to and an Output from AQ11

This appendix contains an example of the program input and output which involves most of the features of the program. Figure B-1 gives the input specification for this example. Figure B-2 gives the output which was obtained. The first page of output repeats the input in a slightly extended form. The next pages show formulas which were generated [in which variables x_1, x_2, x_3, x_4 are substituted by their names, and defined in the input (item P in Fig. B-1)] and the results of the evaluation of formulas on testing events.

Explanation of Figure B-1.

The example involves four variables (NV=4; see item B in Figure B-1(a)), which can take 2, 3, 4 and 2 values, respectively (item E). All variables are nominal, except variable x_3, which is interval (item D). There are 2 classes (NCL=2; item B), each represented by 6 learning events (items F, J). The last event of set (class) 1 is specified as a DVL_1 formula (in the middle of item J). Item H defines the percentage of learning events to be used in each iteration (pass). The restriction on event space is given by a VL_1 decision rule (item I). There are 0 initial hypotheses for class 1 and 2 hypotheses for class 2 (item G). Item K (fig. B-1(b)) lists the hypotheses for class 2. The cost of variable 1 for set 1 is specified as 2 (item L): the cost criteria for the selection of complexes (terms) in the synthesis of covers are in the order 1, 2, 3, 9 (1 and 2 by default; 3 and 9 defined by CRIT(3)=3, CRIT(4)=9 in item B). (For the definition of cost of variables and cost criteria see [Larson, Michalski 75]).

Evaluation of the formulas to be generated is requested (//TEST DD*) and sets of test events supplied, 4 events per class (items M, N). A file containing names of each class (set), each variable and each value of the variable is also supplied (items O, P).

Input for Example

```
    // EXEC PGM=ITCN3,REGION=160K,PARM='ISA(23K),REPORT'
    //STEPLIB DD DSN=USER.P0012.ITCN3,UNIT=DISK,DISP=SHR
    //SYSPRINT DD SYSOUT=A
 A  //PLIDUMP DD SYSOUT=A
    //SAVEF DD SYSOUT=B
    //FT06F001 DD SYSOUT=A
    //TEST DD DSN=JIM,UNIT=DISK,DISP=(NEW,DELETE),SPACE=(TRK,(10,10))
    //SYSIN DD *
     NPASS=2          NV=4       MAXSTAR=30     CRIT(3)=3     CRIT(4)=9
     TITLE=3          NEVE=12    ECHO='ERFZ'    TEST='1'B     TRANS='1'B
 B   RESTRICT='1'B    NCL=2      INDEP='1'B     NGE=100       NCRIT=4
     NTAU=4;
    *********************************************************************
 C                                TEST RUN
    *********************************************************************
 D    1     'I'    3
 E    2   3 4   2
 F    6   6
 G    0   2
 H      .5 1
 I      (X1=0) (X2=0) (X3=0) -> (X4=0).
     0   0 0   0
     0   0 2   0
     0   2 0   1
     0   1 1   0
     0   2 2   1
     FORMULA
        (X4=0) (X2=1 2) (X1=0) (X3=1);
 J   0   2 1   1
     0   0 3   0
     1   2 0   0
     1   1 1   0
     1   0 2   1
     1   2 3   0
```

A	JCL
B	Control parameters
C	Title
D	NSPEC, UTYPE
E	Number of levels/variable
F	Number of events/pass
G	Number of formulas/set
H	Fraction of events/pass
I	Restriction
J	Event list (6 events/set)

Figure B-1 (a)

```
    FORMULA
K   (X1=1) (X2=1 2) (X4=0) (X3=0 1);
    FORMULA
    (X1=1) (X3=2 3) ;
L   Z(1,1)=2;
    //TEST DD *
M   4 4
    0 0 1 0
    0 1 1 0
    0 1 3 0
N   0 0 1 1
    1 1 0 1
    1 2 3 1
    0 1 3 0
    0 1 3 1
    //TRAN DD *
O   'ACCEPT'
    'REJECT'
    'NEW'                      variable x1
               'YES'
               'NO'            values
    'COLOR'                    variable x2
               'RED'
               'BLUE'          values
               'ORANGE'
P   'SIZE'                     variable x3
               'SMALL'
               'MEDIUM'        values
               'LARGE'
               'X LARGE'
    'WEIGTH'                   variable x4
               'HEAVY'         values
               'LIGHT'
```

K	Formulas (2 guesses for set 2)
L	Cost of variables (variable 1 has cost 2 for set 1)
M	Number of test events/set
N	List of test events
O	Name of each set
P	Variable names and variable value names

Figure B-1 (b)

Explanation of Figure B-2.

The first part contains an echo of the input (item A). Next (item B) prints the formulas obtained after the first iteration (pass), which used 50% of the input events (first 3 events in both classes; see item H in Fig. B-1). The classes, variables and values of variables are specified by names. Together with each complex (term) a triple of numbers is printed (NEW, IND, COV) (item C), where

NEW - denotes the number of events covered by the given complex and not covered by the previous complexes on the list of complexes generated for this clas

IND - denotes the number of events covered only by the given complex

COV - the total number of events covered by the given complex.

The program also lists the number of times each cost criterion has been evaluated (item D). Item E gives a symbolic specification of the obtained formulas. Next, an extended confusion matrix is printed (item F) as the result of evaluating the obtained formulas for the testing events (item N in Fig. B-1). We can see from the matrix that all testing events of the first class ('ACCEPT') have been misclassified, and all the events of the second class ('REJECT') have been correctly classified.

Item G specifies the number of times each complex in the cover of each class has been satisfied by testing events in the case of correct decisions (second complex, C2, of class D2 correctly classified 3 testing events, and the third one, C3, correctly classified 1 testing event).

Item H specifies the percentage of correct decisions and the indecision ratio for various values of parameter TAU (generally, the higher TAU, the greater is the number of correct decisions, but also the greater is the indecision ratio).

Item I lists the formulas obtained in the second iteration (which used all the learning events), and item J - the corresponding confusion matrix. We can see that this time 50% of testing events of class 1, and 100% of class 2 were correctly classified. Items K and L give the same information as items G and H, respectively, but for the formulas obtained in the second iteration.

```
*************************************************************************************
                                        TEST RUN
*************************************************************************************


    TR='0'B          NPASS=2                NV=4                     MAXSTAR=30
    TITLE=3          NCL=2                  TEST='1'B                MODE='IC'
    STGY=1           INDEP='1'B             GEN='1'B                 ECHO='ERFZ'
    NGE=100          TAU=1.89999E-02        IRK=2                    CPXEV='1'B
    TRANS='1'B       NTAU=4                 TAU_INC=1.99999E-02      ORD='1'B';
     CRIT LIST  1 0.00  2 0.00  3 0.00  9 0.00
       1  I  3
         2 PASSES  0.50 1.00
     NUMBER OF LEVELS / VARIABLE  2 3 4 2      #EX=1
     NUMBER OF EVENTS / CLASS    6   6         SAVE='0'B
     NUMBER OF FORMUAS / CLASS    0   2        PUNY=1.99999E-02
                                               RESTRICT='1'B
                                               RTEST='0'B

      RESTRICTIONS ON EVENT SPACE
          (X1=0) (X2=0) (X3=0) -> (X4=0).

    LIST OF INPUT EVENTS
A      1 0 0 0 0
       2 0 0 2 0
       3 0 2 0 1
       4 0 1 1 0
       5 0 2 2 1
         (X4=0) (X2=1,2) (X1=0) (X3=1);
       7 0 2 1 1
       8 0 0 3 0
       9 1 2 0 0
      10 1 1 1 0
      11 1 0 2 1
      12 1 2 3 0

    INPUT FORMULAS
         (X1=1) (X2=1,2) (X4=0) (X3=0,1);
         (X1=1) (X3=2,3);

    COSTS OF VARIABLES WHICH ARE NOT 1:               Z(1, 1)=       2;

                              TIME FOR INPUT OF DATA   36 CENTISECONDS


                                                       NEW IND COV
    *****COVER OF ACCEPT*****

      CPX  1: (NEW= YES) (SIZE= SMALL)               (   2   2   2)
      CPX  2: (NEW= YES) (SIZE= LARGE)               (   1   1   1)
                                                  C
B   *****COVER OF REJECT*****

      CPX  1: (SIZE= MEDIUM)                         (   1   1   1)
      CPX  2: (SIZE= X LARGE)                        (   1   1   1)
      CPX  3: (NEW= NO)                              (   1   1   1)
              CRIT #                # TIMES EV.
                1                       12
                2                       11
D               3                        4
                9                        4
    TIME FOR THIS PASS            19 CENTISECONDS
```

Figure B-2 (a)

```
     FORMULAS FOR CLASS    1

      CPX    :(X1 =  0) (X3 =  0)

      CPX    :(X1 =  0) (X3 =  2)

     FORMULAS FOR CLASS    2
E
      CPX    :(X3 =  1)

      CPX    :(X3 =  3)

      CPX    :(X1 =  1)
     NUMBER OF EVENTS IN EACH CLASS    6   6

                                       ASSIGNED DECISION
     CORRECT # EVENTS/  TIE UNSP || D 1  D 2
     ASSIGN   # RK1  DEC          ||
     ---------------------------------------------
     ---------------------------------------------
                                   ||
     D  1 ACCEPT                   ||
                                   ||   .50 1.00
                                   ||   .50 1.00
                                   ||   .50 1.00
                                   ||   .50 1.00
                                   ||
             4/  4                 || | 0 |  4
              1.00                 ||   0% 100%
F    ---------------------------------------------
     ---------------------------------------------
                                   ||
     D  2 REJECT                   ||
                                   ||   .50 1.00
                                   ||       1.00
                                   ||   .50 1.00
                                   ||   .50 1.00
                                   ||
             4/  4                 ||   0  | 4 |
              1.00                 ||   0% 100%

     NUMBER OF CORRECT DECISIONS/COMPLEX

                                        COMPLEXES
     EVENT SETS  C 1 C 2 C 3 C 4 C 5 C 6 C 7 C 8 C 9 C10 C11 C12 C13
G
         1           3   1
         2
```

Figure B-2 (b)

```
                         TAU ESTIMATION TABLE  (% CORRECT / INDECISION RATIO)

    CLASS                                       VALUE OF TAU
                   0.00      0.02      0.04      0.06
      1         0.00/1.00 0.00/1.00 0.00/1.00 0.00/1.00
H     2         1.00/1.00 1.00/1.00 1.00/1.00 1.00/1.00

    TOTALS      0.50/1.00 0.50/1.00 0.50/1.00 0.50/1.00

                                                FINAL STATISTICS
                                           INDECISION RATIO:  1.00
                                           PERCENT CORRECT:  50.00
                    TIME TO EVALUATE FORMULAS        35 CENTISECONDS


    CPX  1:(NEW= YES) (SIZE= SMALL MEDIUM) (WEIGHT= HEAVY)      (   3   2   3)

    CPX  2:(NEW= YES) (SIZE= LARGE)                             (   2   2   2)

    CPX  3:(NEW= YES) (SIZE= SMALL)                             (   1   1   2)

I   *****COVER OF REJECT*****

    CPX  1:(SIZE= MEDIUM) (WEIGHT= LIGHT)                       (   1   1   1)

    CPX  2:(SIZE= X LARGE)                                      (   2   1   2)

    CPX  3:(NEW= NO)                                            (   3   3   4)

                                   CRIT #        # TIMES EV.
                                     1               8
                                     2               7
                                     3               4
                                     9               3

                   TIME FOR THIS PASS         20 CENTISECONDS

                                      ASSIGNED DECISION
    CORRECT # EVENTS/  TIE UNSP || D 1  D 2
    ASSIGN   # RK1  DEC         ||
     -----------------------------------------
     -----------------------------------------
                                ||
    D  1 ACCEPT                 || 1.00  .50
                                || 1.00  .50
                                ||  .67 1.00
                                ||  .67 1.00
                                ||
J              4/  4            || | 2 |  2
               1.00             ||  50%  50%
     -----------------------------------------
     -----------------------------------------
    D  2 REJECT                 ||  .50 1.00
                                ||      1.00
                                ||  .67 1.00
                                ||  .50 1.00
                                ||
               4/  4            ||   0  | 4 |
               1.00             ||   0% 100%
```

Figure B-2 (c)

```
                         NUMBER OF CORRECT DECISIONS/COMPLEX

                                    COMPLEXES
      EVENT SETS    C 1 C 2 C 3 C 4 C 5 C 6 C 7 C 8 C 9 C10 C11 C12 C13
K
          1           2
          2               3   1



                             TAU ESTIMATION TABLE   (% CORRECT / INDECISION RATIO)
      CLASS
                                                    VALUE OF TAU
                         0.00      0.02      0.04      0.06
L         1           0.50/1.00 0.50/1.00 0.50/1.00 0.50/1.00
          2           1.00/1.00 1.00/1.00 1.00/1.00 1.00/1.00

      TOTALS          0.75/1.00 0.75/1.00 0.75/1.00 0.75/1.00

                                            FINAL STATISTICS
                                        INDECISION RATIO:  1.00
                                        PERCENT CORRECT:  75.00


                         TIME TO EVALUATE FORMULAS      28 CENTISECONDS
```

Figure B-2 (d)

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-78-867	2.	3. Recipient's Accession No.
4. Title and Subtitle SELECTION OF MOST REPRESENTATIVE TRAINING EXAMPLES AND INCREMENTAL GENERATION OF VL_1 HYPOTHESES: the underlying methodology and the description of programs ESEL and AQ11			5. Report Date May 1978
			6.
7. Author(s) R. S. Michalski and J. B. Larson			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, IL 61801			10. Project/Task/Work Unit No.
			11. Contract/Grant No. NSF MCS 76-22940
12. Sponsoring Organization Name and Address National Science Foundation Washington, D.C.			13. Type of Report & Period Covered
			14.
15. Supplementary Notes			

16. Abstracts

The paper describes the underlying theoretical framework and operational details of two programs, ESEL and AQ11, for computer induction within the framework of the variable-valued logic system VL_1 (i.e., a statement calculus which involves variables with an arbitrary number of discrete values [Michalski 1974]):

ESEL - A supporting program for selecting 'most representative' learning and/or testing VL_1 events from a large data base of events. The programs provides the input to the program AQ11.

AQ11 - A program for incremental generation of VL_1 hypotheses, which are generalized and optimized descriptions of input event sets. The program also provides a facility for evaluating the performance of these inferred hypotheses on testing events.

Given a large set of examples describing certain objects or situations, program ESEL selects from them a small subset of the most representative ones. The examples have to be in the form of VL_1 events, i.e., in the form sequences of values of certain discrete variables (or descriptors). In selecting the events, the program distinguishes among three types of descriptors: nominal descriptors, whose value set is an unordered set, linear descriptors, whose value set is a linearly ordered set, and structured descriptors, whose value set is a tree-ordered set.

Events selected by ESEL are input to program AQ11, which generates VL_1 hypotheses describing the events. The program can work incrementally, i.e., given a working hypothesis (a set of rules) obtained at some stage, and a set of events, the program can modify the hypothesis to make it consistent with the events.

Program AQ11 also has the facility to test the performance of a given hypothesis on a set of testing events, and to compute an extended confusion matrix.

17. Key Words and Document Analysis. 17a. Descriptors

VL_1, Variable-valued logic, inductive inference, incremental hypothesis generation, incremental induction, hypothesis modification, rulebased deduction, deductive inference

18. Availability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages
	20. Security Class (This Page) UNCLASSIFIED	22. Price