83-11

Reports

of the

Intelligent

Systems

Group

INCREMENTAL GENERATION OF VL1 HYPOTHESES:
the underlying methodology and the description of program AQ11

by

R. S. Michalski and J. B. Larson

Revised by K. Chen

January 1983

File No. UIUCDCS-F-83-905

ISG 83-5

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN   URBANA, ILLINOIS

Date: January 5, 1983
Subject: A simple description of the AQ11p files archived
on a magnetic tape (1600 bpi).

1) Files on the tape

aq11.obj : the object code of AQ11p program, version 1/5/83
aq11 : a command file to run the AQ11p program
filter : the output format of aq11.obj was intended for printers that take the first character of each line as control signal, for those printers that do not work this way, this program can be used to make the translation.
testf : contains the testing events for aq11ex1
transf : contains the symbolic names of classes, variables and variable values for aq11ex1
aq11ex1 : the input data for the first AQ11p example
manual.tape: this manual
aq11.source: the source code (written in PASCAL) of AQ11p

2) How to list the content of the tape

```
tar t0
```

3) How to transfer the files on the tape to disk

```
tar xv0
```

4) How to run AQ11p

```
aq11 input-file output-file
```

5) To compile aq11.source: eihter

```
       pi aq11.source > error-listing
or     pc aq11.source > error-listing
```

does the job. The default file names for the object code produced are ´a.out´ for ´pc´, and ´obj´ for ´pi´. For faster compilation, use ´pi´. For faster execution, use ´pc´.

Incremental Generation of VL_1 Hypotheses: The Underlying Methodology and the Description of Program AQ11

R. S. Michalski and J. B. Larson

MLI 83-11
ISG 83-5
UIUCDCS-F-83-905

83-11

INCREMENTAL GENERATION OF VL1 HYPOTHESES: the underlying methodology and the description of program AQ11

by

R. S. Michalski and J. B. Larson

Revised by K. Chen
January 1983

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

ISG 83-5

File No. UIUCDCS-F-83-905

This work was supported in part by the National Science Foundation under Grants MCS 76-22940, MCS 82-05166 and MCS 82-05896, and the Office of Naval Research under Grant N00014-82-K-0186.

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

1.1 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, andhave 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 this framework is too restricted to model many "real life" hypothesis formation processes, it is still sufficiently rich to provide a challenging topic and a basis for developing inductive learning techniques applicable to a variety of practical problems.

Hypotheses are expressed here as (constant-free) disjunctive normal VL_1 expressions (DVL_1 expressions*). A DVL_1 expression is a disjunction of complexes,** where a complex 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 $= \neq \geqslant \leqslant$

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.

**The "complex" replaces the name "term" in [Michalski 74].

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

$[x_i = 1]$ value of x_i is equal to 1?

$[x_i = 1v3]$ value of x_i is equal to 1 or 3?

$[x_i = 1..3]$ value of x_i is between 1 and 3, inclusively?

An example of a complex:

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

The above complex 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:

$$T_1 \quad V \quad T_2 \quad V \quad T_3$$

where T_1, T_2, T_3 are complexes. The formula is satisfied if complex 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.

1.2 <u>Description of Methodology</u>

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 <u>cover</u>, $C(E_1/E_o)$, of an <u>event set</u> E_1 <u>against</u> the <u>event set</u> 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:

<u>Step 1</u>. 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 <u>be covered</u> 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*:

$$F_i^+ = F_i \setminus \tilde{V}_i \qquad (8)$$

$$F_{ij}^- = \tilde{V}_i \cap F_j,\ j=1,2,\ldots,m;\ j \neq i \qquad (9)$$

(see Figure 1).

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.

<u>Step 2</u>. 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 $\tilde{V}_i \cup F_i^+$, i=1,2,...,m:

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

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

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

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

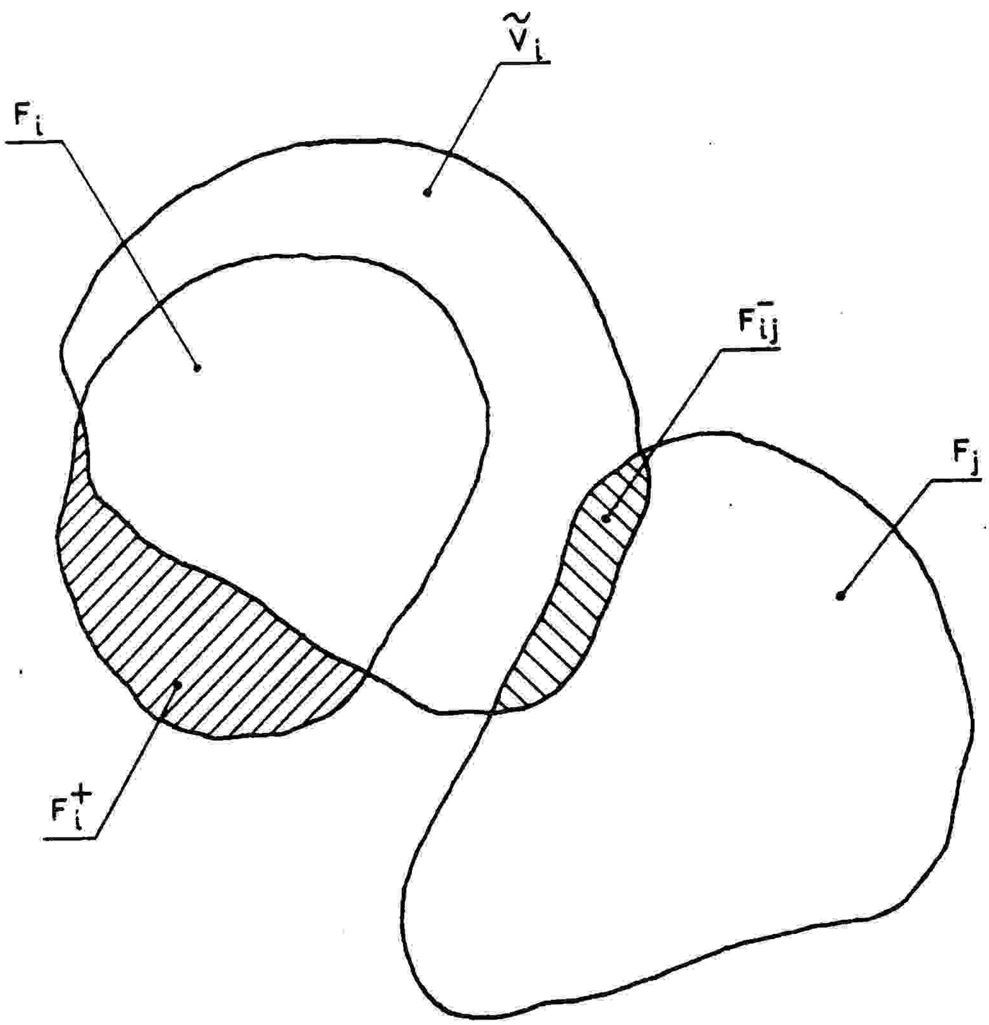


Illustration of sets F_i^+ and F_{ij}^-

Figure 1

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:

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

(The point is that directly simplifying a union of complex is difficult; but 'substracting' a complex from a complex 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 complexes. Some complexes in a V_i^1 may represent (cover) only a few events in F_i. Such 'low weight' complexes are replaced by the events (facts) themselves (since an event takes less memory than a complex). In program AQ11, parameter PUNY specifies the minimum percent of events which a complex has to cover to be a 'high weight' complex.

For example, if puny = 0.02, and a set F_i has 100 events, then all complexes which cover 3 or more events ($3 > 0.02 \times 100$) are 'high weight' complexes. Complexes which cover 1 or 2 events are replaced with those events.

1.3 An alternative way of handling exception events

In the procedure above, the exception events were represented by complexes 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 complex T (in a given formula) is done by logically multiplying the complex by the negation of the event:

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

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

1.4 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. 14), this may not be the best approach since only a few complexes in each intermediate set 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

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

1.5 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 Decision)	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 complex (a product of selectors),

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

$DC(\{V_i\}, e)$ - 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.*

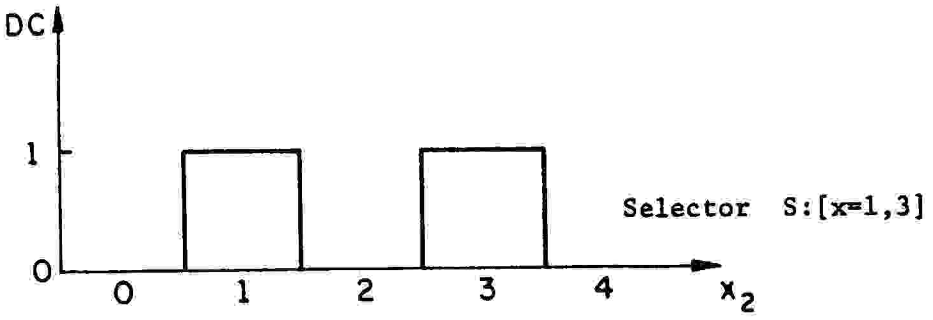
The basic definition of the degree of consonance, DC(S,e), of a selector comes from the evaluation rules in VL_1 [Michalski 74]. Assuming that the output domain of the formulas D = {0,1} we have:

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

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. 2).

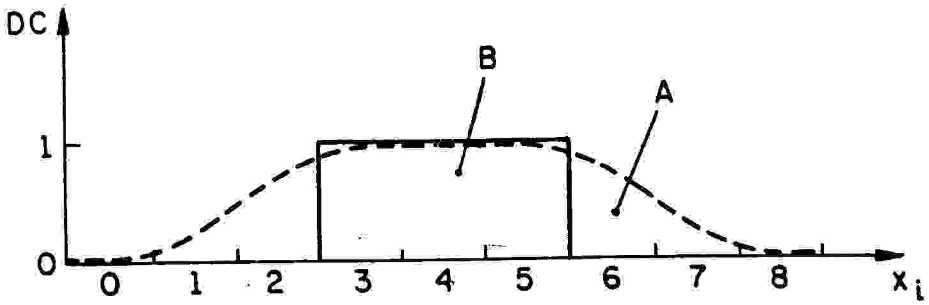
Alternative evaluation schemes can take into consideration the structure of the domainof the variable in the selector.If a variable is linear, it seems that the above definition of DC(e,S) is too rigid. For example, if a linear variable $x_i=13$ and S: $[x_i=14..18]$, the selector is evaluated to 0, 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. 3).

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



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

Figure 2



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

Figure 3

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

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. 4 a and b).

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

In the definition of VL_1, the consonance degree of a complex was defined as the minimum of values of selectors in the term. In AQ11, the consonance degree of a complex is computed as the ratio of the number of selectors satisfied in the complex to the total number of selectors in the complex. If all selectors in a complex are satisfied, then both definitions give the same value. If this is not the case, the latter definition differentiates between the complexes 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 complex into the arithmetic product of DC-s of selectors.

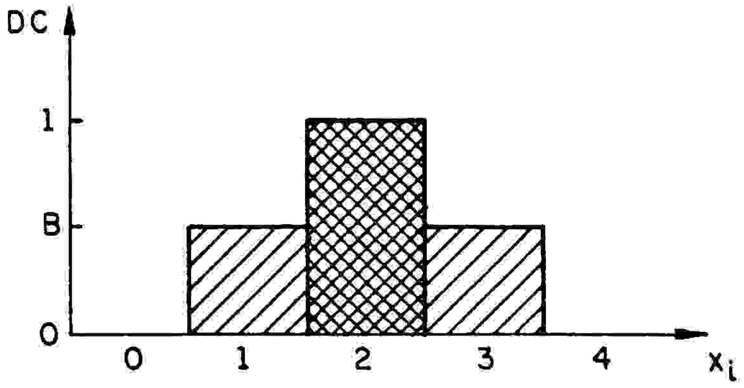
(3) *Definition of the degree of consonance of a formula.*

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 complexes T_i, in the formula (i.e., as defined in VL_1), i.e.:

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

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

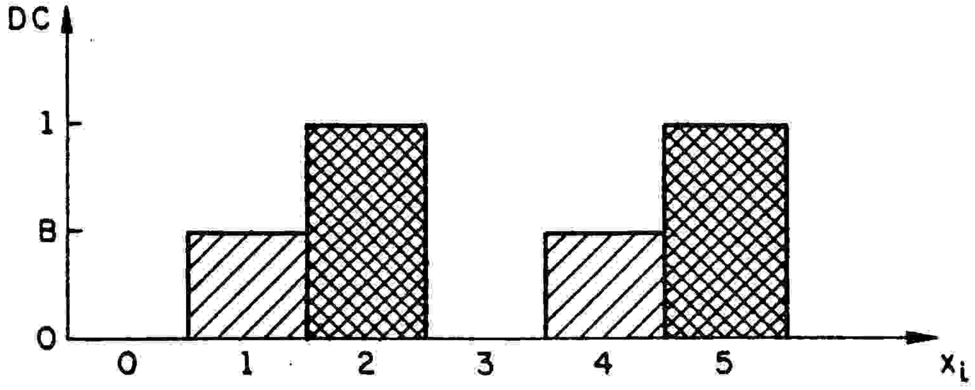
It is useful sometimes to generate more than one formula describing a given class. The reason is that having more than one formula per class may



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 4

improve the reliability of decision making. In AQ11 we have accepted the convention 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 1.5) 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 1.5).

For each testing event, values of DC of ranked decisions are printed by the program AQ11 as rows in a generalized confusion matrix (see Fig. 5 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 matrix also contains a column #Rank1dec/# Events 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 Unsp=*). Figure 5 gives an example of such a confusion matrix. The matrix was computed for 3 classes D1, D2 and D3 described by formulas:

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

D2: [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 Decision	#Rank1 dec/ Event	#Ties	Unsp	D1	D2	D3
D1				.68	.50	.50
		2		1.00	.50	1.00
			*			0.50
Total	4/3 1.33	2	1	2 67%	0 0%	1 33%

Underscored numbers indicate rank1 correct decisions.

An example of an extended confusion matrix

Figure 5

The value of #Rank1dec/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.

1.6 Program user's guide

The program requires two sets of input parameters: control parameters and data parameters. All but 3 control parameters have default values, therefore only (neve, 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.

1.6.1 Control parameters

- nv (no default value)

  Example: nv = 30

  Possible values: integer in the range 1:30

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

- ncl (no default value)

  Example: ncl = 50

  Possible values: 1:50

  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

  The maxstar is the maximum number of complexes 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: integer in the range 1:20

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.

error

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 transf mustbe given to the program (see transf below).

- puny

Example: puny = .020 (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]

The range of the equivalence class of consonance degree in computing the confusion matrix. Any two values (degrees of consonance) within tau value from 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 (default 0.019).

- 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 crit 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 complex, and not covered by previous complexes

2. Minimize number of selectors

3. Minimize cost of all variables in the complex. 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 complex

- #ex

Example: #ex = 1 (default value)

Possible values: positive integer

During some phase of the program, exception complexes are formed (description of events which are covered by hypotheses but should not have been). #ex gives the number of redundant exception complexes (i.e., the complexes 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 complexes and the size of the sets F^+ and F^- described in Section 1.1.

- stgy

Example: stgy = 1 (default value)

Possible values: 1 or 2

If stgy has the value 1, then exception complexes 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 lines 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 complexes 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. If 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 complex 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 complex is created from the generated term which has the following properties):

a. The new complex covers the same events.

b. The new complex contains the same variables.

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

- echo

Example: echo = 'erz' (default value)

Possible values: A string containing any of the characters {z,e,r,f,p,t}

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

e	:	Events
r	:	Restrictions
z	:	Variable costs
f	:	Input (initial hypotheses)
p	:	Parameters
t	:	Translation file

The default echos events, restrictions and variable costs if they are in the input. This parameter should be put before any other parameters.

- tolerance(J)

  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.

- ntau

  Example: ntau = 0 (default value)

  Possible values: integer in [0:8]

  This parameter, if not zero, specifies the number of evaluations using a different value of tau parameter (defining the equivalence class of the degree of consonance in evaluating the testing events). The so called "tau estimation table" gives the percentages of correct decisions for each class in the evaluation procedure using ntau values of tau beginning at 0 with increments of tauinc. If ntau=0, no "tau estimation table" will be printed.

- tauinc

  Example: tauinc = .02 (default value)

  Possible values: Real in [0:1]

  This is the increment used in the tau estimation table.

- neve (no default value)

  Example: neve = 2000

  Possible values: integer in the range 1:2000

  This parameter specifies the number of events in the training set.

- nge

  Example: nge = 200

  Possible values: any possible integer

  This parameter specifies the initial storage for complexes.

- Semicolon (;): This must be entered to terminate the control parameters.

1.6.2 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.)

- titles

  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 type 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 [1:20]

  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 initial conjunctive hypotheses (complexes) 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 that case, this parameter must be omitted)

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 semicolons 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.

- specification of event sets

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 (# of classes) lists of initial formulas (hypotheses), each having nf complexes.

- z

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

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

- testf

  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.

- transf

  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:

```
'TEMPERATURE'

        'COLD'
        'MODERATE'
        'WARM'

'HUMIDITY'

         'DAMP'  'DRY'
```

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.

1.6.4 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/EO) is being created, then the event of EO is ignored.

The output from the evaluation part of the program consists of an extended confusion matrix, as described in section 1.5.

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 ntau is not zero, then tau estimation table is printed, giving the indecision ratio and number of correct decisions for each class for ntau values of tau, beginning with 0 in increments of tauinc.

2. SUMMARY

We have described here the underlying methodology and computer programs for incrementally generating VL_1 hypotheses for given event sets and then automatically testing them on the supplied testing events. This program can be used for making experiments in induction of descriptions from examples in various applied fields.

ACKNOWLEDGMENT

The research on the revised version of AQ11 program described here was supported in part by National Science Foundation Grants MCS 82-05166 and MCS 82-05896, and the Office of Naval Research Grant N00014-82-K-0186.

The research on the original implementation was supported in part by the National Science Foundation Grant MCS 76-22940 and in part by a Senior Visiting Fellowship from the Science Research Council in U.K.

The initial version of this 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 the University of Essex, 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 this paper.

REFERENCES

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.

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

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

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.

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.

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.

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

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.

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.

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.

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

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

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

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

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. To execute AQ11 on UNIX operating system, four files must exist in the working directory:

aq11.obj — the object code of the AQ11 program.

aq11 — an executable file which contains the following shell command:

```
cat $1 | aq11.obj > $2
```

transf — contains the names of all variables and variable values.

testf — contains a list of events to be tested against the output hypotheses of AQ11.

The following command can then be issued to execute AQ11:

```
aq11 inputfile outputfile
```

where inputfile is the input data file to the AQ11 program, and outputfile is the result produced by the AQ11 program.

Note that since the output format of the program ´aq11.obj´ was intended for a traditional line printer which takes the first character of each line as a control signal, for those printers that do not work this way, a program ´filter´ is available to make the proper translation. The content of the command file ´aq11´ should be change to

```
cat $1 | aq11.obj | filter >$2
```

and the file ´filter´ must reside in the working directory.

2. Control parameters

Parameter	Default	Description
nv	---	Number of variables
neve	---	Total number of training events
ncl	---	Number of classes
mode	'ic'	Mode of operation, one of {'ic','dc','vl'}
maxstar	10	Maximum star size
echo	'erz'	Echo input, combination of {'e','v','z','f','p','t'}
ncrit	2	Number of criteria
crit(1)	1	Criterion 1
crit(2)	2	Criterion 2
crit(3)	3	Criterion 3
crit(4)	5	Criterion 4
crit(5)	9	Criterion 5
title	0	Number of lines in title
restrict	'0'b	Accept restrictions
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 transf 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
ntau	0	Number of columns in 'tau' estimation table
tauinc	.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

3. Data parameters

Parameter	Description
title	Lines of title (if any)
nspec	Number of variables for which type "type" is specified
type	The type of these variables. 'i' (linear) or 'f' (nominal)

vspec	Indices of variables of type defined by "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
rest	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 variables terminated with semi-colon

5. Files

	Description
transf	If transf is '1'b, then this file be created, containing names of classes, variables and variable values. Each Name must be in quotes.
testf	If testf is '1'b, then this file must be created, containing testing events.

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 $\bar{n}v=4$; see item A in Figure B-1(a)), which can take 2, 3, 4 and 2 values, respectively (item B). All variables are nominal, except variable x_3, which is interval (item C). There are 2 classes (ncl=2; item A), each represented by 6 learning events (items E, I). The last event of set (class) 1 is specified as a DVL_1 formula (in the middle of item I). Item G 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 H). There are 0 initial hypotheses for class 1 and 2 hypotheses for class 2 (item F). Item J (fig. B-1(b)) lists the hypotheses for class 2, The cost of variable 1 for set 1 is specified as 2 (item K): 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 A). (For the definition of cost of variables and cost criteria see [Larson, Michalski 75]).

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

```
    echo=´ezrfpt´     npass=2    nv=4        maxstar=30   crit(3)=3    crit(4)=9
    title=3           neve=12    test=´1´b   trans=´1´b
A   restrict=´1´b     ncl=2      indep=´1´b  nge=100
    ncrit=4           ntau=4;
    ******************************************************************
B                     test run:  pickup truck
    ******************************************************************
C   1   ´i´   3
D   2  3 4  2
E   6  6
F   0  2
G    0.5  1.0
H   (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);
I   0  2 1  1
    0  0 3  0
    1  2 0  0
    1  1 1  0
    1  0 2  1
    1  2 3  0
    formula
J   (x1=1)(x2=1  2)(x4=0)(x3=0  1);
    formula
    (x1=1)(x3=2  3);
K   z(1,1)=2;
```

A Control parameters

B Titles

C nspec, vtype, type. That is, there is 1 variable, x3, which is a linear (´i´) variable.

D Number of levels/variable. There are 2 levels for x1, 3 levels for x2,, 4 levels for x3, and 2 levels for x4.

E Number of events/pass. There are 6 events for the first pass, and another 6 events for the second pass.

F Number of initial hypotheses/set. There are 2 initial hypotheses for event set 2, but none for the first event set.

G Fraction of events/pass. 50% of input events are processed in the first pass, and another 50% for the second pass.

H Restriction.

I Event list (6 events/event set).

J Formulas (2 guesses for set 2).

K Cost of variables (variable 1 has cost 2 for set 1).

Figure B-1 (a)

Input file ´testf´:

```
L    4 4
    0 0 1 0
    0 1 2 1
    0 2 0 0
    0 2 2 0
M   1 1 0 1
    1 2 3 1
    1 0 2 0
    0 1 3 1
```

Input file ´transf´:

```
N   ´accept´
    ´reject´
    ´new´
      ´yes´
      ´no´
    ´color´
      ´red´
      ´blue´
      ´orange´
O   ´size´
      ´small´
      ´medium´
      ´large´
      ´extra large´
    ´weight´
      ´heavy´
      ´light´
```

L Number of test events/set. There are 4 testing events assumed to be of event set 1, and another 4 to be of event set 2.

M List of testing events.

N Name of each set. The name of the first event set is ´accept´, and the name of the second event set is ´reject´.

O Variable names and variable value names. The four variables are ´new´, ´color´, ´size´ and ´weight´. The three possible values for the variable ´color´ are ´red´, ´blue´ and ´orange´ respectively, etc.

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 G 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

[newly covered, indep covered, totally covered]

newly covered - 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 class

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

totally covered - 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 C). Next, an extended confusion matrix is printed (item D) as the result of evaluating the obtained formulas for the testing events (item M in Fig. B-1). We can see from the matrix that 25% of testing events of the first class ('accept') have been misclassified, and all the events of the second class ('reject') have been correctly classified.

Item E 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 (first complex, CI, of class 1 correctly classified 1 testing events, and the second one, C2, correctly classified 2 testing event, etc.).

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

```
AQ11p - 2.0 (Pascal version): multi-step rule induction and refinement

  Initialization time:                 0.134   seconds

  Echo of non-default parameters:

     npass=        2
     nv=           4
     maxstar=      30
     crit(3)=      3
     crit(4)=      9
     title=        3
     neve=         12
     test=         ´1´b
     trans=        ´1´b
     restrict=     ´1´b
     ncl=          2
     indep=        ´1´b
     nge=          100
     ncrit=        4
     ntau=         4

   All parameters for this run:

      Number of classes (ncl):                                 2
A     Number of variables (nv):                                4
      Number of events (neve):                                12
      Maximum star size (maxstar):                            30
      Allowable # of exceptions (numex):                       1
      Number of passes (npass):                                2
      Number of columns in the tau table (ntau):               4
      Increment of tau in the tau table (tauinc):              0.020
      Ignored difference in consonance degrees (tau):          0.019
      Highest rank of consonance degree to be printed (irk):   2
      Ignored event coverage for the next pass (puny):         0.020
      Initial storage for complexes (nge):                   100
      Mode of cover synthesis:  Intersecting covers (ic).

    Information to be echoed :
      Events?        yes
      Costs?         yes
      Restrictions?  yes
      Formulas?      yes
      Parameters?    yes
      Translation?   yes

    Number of criteria (ncrit):  4
                              (1)  (2)  (3)  (4)  (5)  (6)
      Criterion:               1    2    3    9    0    0
      Tolerances:             0.00 0.00 0.00 0.00 0.00 0.00

      Strategy:               1
```

Figure B-2 (a)

A

```
Test:      test complexes?      yes
Ord:       order complexes?     yes
Cpxev:     evaluate complexes?  yes
Indep:     print independents?  yes
Tr:        trace formulation?   no
Startr:    trace star formatn?  no
Restrict: input restrictions?  yes
Opt:       print # cost funcs?  yes
Gen:       trim cpxs for outpt? yes
Rtest:     apply rstrs to test? yes
Trans:     translate output?    yes

*****************************************************************
                test run:  pickup truck
*****************************************************************

Event specifications
  Variable:                1      2      3      4
  Type:                    nom    nom    lin    nom
  Number of values:        2      3      4      2

Class:                                    1      2
The number of events in the class:        6      6
The number of formulas in the class:      0      2

Percentage of events to use per pass:     0.50   1.00
Echo of restriction: (x 1=  0)(x 2=  0)(x 3=  0)
                     implies
                     (x 1=)(x 2=)(x 3=)(x 4=  0)


Echo of input events:
     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,
     6:   (x 1=  0)(x 2=  1 v  2)(x 3=  1)(x 4=  0)
     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,

Echo of input initial hypothesis:

  Class   1: no formula

  Class   2: Cpx  1:(x 1=  1)(x 2=  1 v  2)(x 3=  0 v  1)(x 4=  0)
             Cpx  2:(x 1=  1)(x 3=  2 v  3)

Costs of variables different from 1:
  z( 1, 1) =  2
```

Figure B-2 (b)

```
     Translation file:
       Class   1: accept
       Class   2: reject

          Variable x 1:  new
                 Value= 0:  yes
                 Value= 1:  no

          Variable x 2:  color
                 Value= 0:  red
                 Value= 1:  blue
                 Value= 2:  orange

          Variable x 3:  size
                 Value= 0:  small
                 Value= 1:  medium
                 Value= 2:  large
                 Value= 3:  extralarge

          Variable x 4:  weight
                 Value= 0:  heavy
A                Value= 1:  light

     Time of reading input data:        0.916  seconds
     Results of pass 1.

     * * * Initial hypotheses for each class * * *

       Class  1 "accept" :   no formula

       Class  2 "reject" :   Cpx 1: (new=no)(size=large..extralarge)
              [ newly covered=   0, indep covered=   0, totally covered=    0 ]

                             Cpx 2: (new=no)(color=blue,orange)
                                    (size=small..medium)(weight=heavy)
              [ newly covered=   0, indep covered=   0, totally covered=    0 ]


      Number of events in each class:
        Class  1 "accept" :    3
        Class  2 "reject" :    3

      * * * Generated hypothesis for class  1 "accept" from 3 training events * * *

        Cpx 1: (new=yes)(size=small)
               [ newly covered=   2, indep covered=   2, totally covered=   2 ]

B       Cpx 2: (new=yes)(size=large)
               [ newly covered=   1, indep covered=   1, totally covered=   1 ]
```

Figure B-2 (c)

```
   * * * Generated hypothesis for class  2 "reject" from 3 training events * * *

     Cpx 1: (size=medium)
            [ newly covered=   1, indep covered=   1, totally covered=   1 ]

B    Cpx 2: (size=extralarge)
            [ newly covered=   1, indep covered=   1, totally covered=   1 ]

     Cpx 3: (new=no)
            [ newly covered=   1, indep covered=   1, totally covered=   1 ]

   Criterion # 1 evaluated 12 times.
C  Criterion # 2 evaluated 11 times.
   Criterion # 3 evaluated 4 times.
   Criterion # 9 evaluated 4 times.
                                           Assigned decision

   Correct    #Rank1 dec/   #Ties   Unsp!!  D 1  D 2
   Decision   #Events                   !!
   ----------------------------------------------------
   ----------------------------------------------------
                                        !!
                                        !!  .50 1.00
   D 1                                  !! 1.00  --
   accept                               !! 1.00  --
                                        !! 1.00  --
   ----------------------------------------------------
                 4/  4                  !!   3    1
D  Total         1.00                   !!  75%  25%
   ----------------------------------------------------
   ----------------------------------------------------
                                        !!
                                        !!  .50 1.00
   D 2                                  !!  --  1.00
   reject                               !!  .50 1.00
                                        !!  .50 1.00
   ----------------------------------------------------
                 4/  4                  !!   0    4
   Total         1.00                   !!   0%  100%

   Note: a rank 1 decision which is correct is underlined.
         "--" is equivalent to "0.00".

   Number of testing events satisfying individual complexes
     in the correct class description.

                          Complexes

                          C 1  C 2  C 3
E   Class 1 "accept"       1    2
    Class 2 "reject"       0    2    2

   Note: The entries (classx, Ci) in the above table show the number
         of testing events of  calssx ( as indicated by the file
         "testf") that were covered by the ith complex (Ci).
```

Figure B-2 (d)

```
     Tau estimation table  (the percentage of correct decision / indecision ratio)

                                     Value of tau
                               0.00          0.02          0.04          0.06
       Class 1 "accept"     0.75/ 1.00    0.75/ 1.00    0.75/ 1.00    0.75/ 1.00
       Class 2 "reject"     1.00/ 1.00    1.00/ 1.00    1.00/ 1.00    1.00/ 1.00
F
       Totals:              0.88/ 1.00    0.88/ 1.00    0.88/ 1.00    0.88/ 1.00

     Final statistics:
       Indecision ratio:        1.00
       Total percent correct:  87.50


     Computing time for this pass:      0.434   seconds
     Results of pass 2.

     * * * Initial hypotheses for each class * * *

       Class   1 "accept"  :   Cpx 1: (new=yes)(size=small)
               [ newly covered=   2, indep covered=   2, totally covered=   2 ]

                               Cpx 2: (new=yes)(size=large)
               [ newly covered=   1, indep covered=   1, totally covered=   1 ]

       Class   2 "reject"  :   Cpx 1: (size=medium)
               [ newly covered=   1, indep covered=   1, totally covered=   1 ]

                               Cpx 2: (size=extralarge)
               [ newly covered=   1, indep covered=   1, totally covered=   1 ]

                               Cpx 3: (new=no)
               [ newly covered=   1, indep covered=   1, totally covered=   1 ]


     Number of events in each class:
       Class   1 "accept"  :    6
       Class   2 "reject"  :    6


     * * * Generated hypothesis for class  1 "accept"  from 6 training events * * *

        Cpx 1: (new=yes)(size=small..medium)(weight=heavy)
               [ newly covered=   3, indep covered=   2, totally covered=   3 ]

G       Cpx 2: (new=yes)(size=large)
               [ newly covered=   2, indep covered=   2, totally covered=   2 ]

        Cpx 3: (new=yes)(size=small)
               [ newly covered=   1, indep covered=   1, totally covered=   2 ]
```

Figure B-2 (e)

```
     * * * Generated hypothesis for class  2 "reject" from 6 training events * * *

        Cpx 1: (size=medium)(weight=light)
                [ newly covered=   1, indep covered=   1, totally covered=   1 ]

G       Cpx 2: (size=extralarge)
                [ newly covered=   2, indep covered=   1, totally covered=   2 ]

        Cpx 3: (new=no)
                [ newly covered=   3, indep covered=   3, totally covered=   4 ]


     Criterion # 1 evaluated 8 times.
     Criterion # 2 evaluated 7 times.
     Criterion # 3 evaluated 4 times.
     Criterion # 9 evaluated 3 times.
                                                      Assigned decision

     Correct     #Rankl dec/    #Ties    Unsp!!  D 1   D 2
     Decision    #Events                   !!
     --------------------------------------------------
     --------------------------------------------------
                                           !!
                                           !! 1.00   .50
     D 1                                   !! 1.00   .50
     accept                                !! 1.00   --
                                           !! 1.00   --
     --------------------------------------------------
                   4/  4                   !!    4     0
H    Total         1.00                    !! 100%    0%
     --------------------------------------------------
     --------------------------------------------------
                                           !!
                                           !!  .50  1.00
     D 2                                   !!  --   1.00
     reject                                !!  .50  1.00
                                           !!  .50  1.00
     --------------------------------------------------
                   4/  4                   !!    0     4
     Total         1.00                    !!   0%  100%


     Note: a rank 1 decision which is correct is underlined.
           "--" is equivalent to "0.00".
     Number of testing events satisfying individual complexes
       in the correct class description.

                                 Complexes

                                 C 1  C 2  C 3
I      Class 1 "accept"           2    2    0
       Class 2 "reject"           0    2    2

     Note: The entries (classx, Ci) in the above table show the number
           of testing events of  calssx ( as indicated by the file
           "testf") that were covered by the ith complex (Ci).
```

Figure B-2 (f)

```
     Tau estimation table  (the percentage of correct decision / indecision ratio)

                                      Value of tau
                                  0.00          0.02          0.04          0.06
       Class 1 "accept"        1.00/ 1.00    1.00/ 1.00    1.00/ 1.00    1.00/ 1.00
       Class 2 "reject"        1.00/ 1.00    1.00/ 1.00    1.00/ 1.00    1.00/ 1.00

J      Totals:                 1.00/ 1.00    1.00/ 1.00    1.00/ 1.00    1.00/ 1.00

     Final statistics:
       Indecision ratio:          1.00
       Total percent correct:  100.00


     Computing time for this pass:       0.366   seconds

     Maximum size of complex storage was:          100
```

Figure B-2 (g)

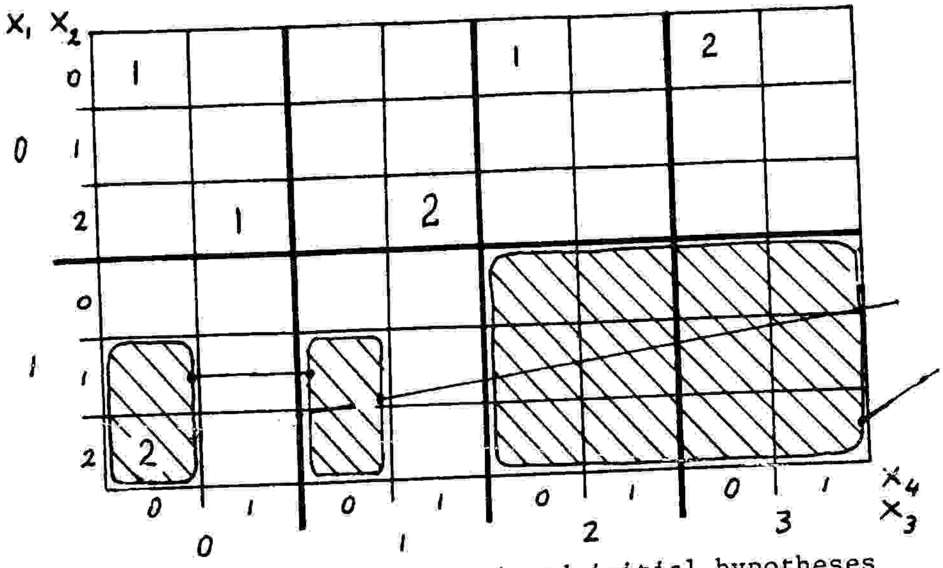
Explanation of Figure B-3, B-4 and B-5

Figure B-3, B-4 and B-5 are graphical illustrations of how the output hypotheses of each pass improve as more training events are available.

Figure B-3: shows the learning events and initial hypotheses

Figure B-4: shows the intermediate hypotheses (after pass 1)

Figure B-5: shows the final hypotheses (after pass 2)



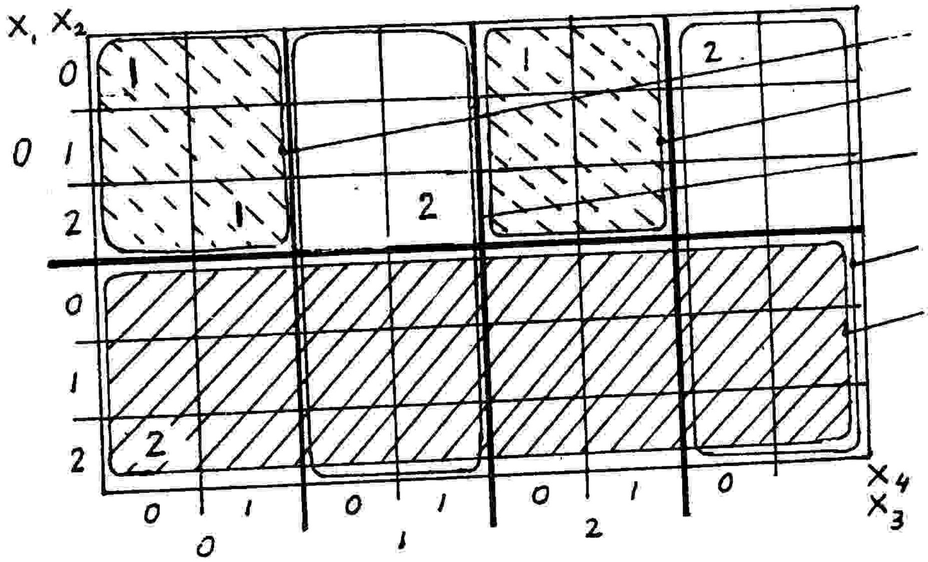
No initial hypothesis
for class 1

Initial hypothesis
for class 2

cpx1: $[x_1=1][x_2\geq1][x_3\leq1][x_4=0]$

cpx2: $[x_1=1][x_3\geq2]$

Learning events for pass 1 and initial hypotheses
Figure B-3



Class 1

cpx1: $[x_1=0][x_3=0]$

cpx2: $[x_1=0][x_3=2]$

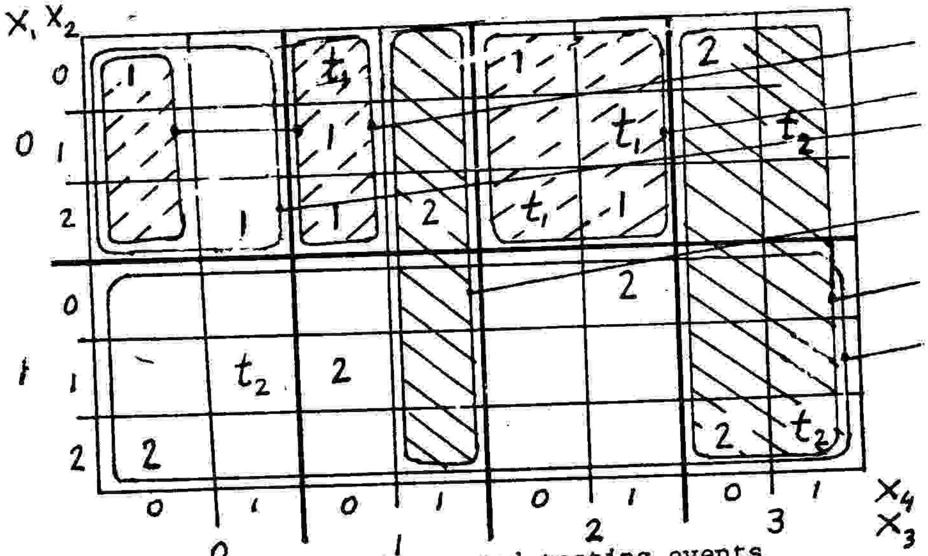
Class 2

cpx1: $[x_3=1]$

cpx2: $[x_3=3]$

cpx3: $[x_1=1]$

Hypotheses after pass 1
Figure B-4



Class 1

cpx1: $[x_1=0][x_3\leq1][x_4=0]$

cpx2: $[x_1=0][x_3=2]$

cpx3: $[x_1=0][x_3=0]$

Class 2

cpx1: $[x_3=1][x_4=1]$

cpx2: $[x_3=3]$

cpx3: $[x_1=1]$

Final hypotheses and testing events
Figure B-5