AQVAL/1 (AQ7) USER'S GUIDE
AND PROGRAM DESCRIPTION

by

Ryszard S. Michalski
James Larson

Report No. 731, Department of Computer Science, University of Illinois,
AQUAL/1 (AQ7)
User's Guide and Program Description

by

James Larson
R. S. Michalski

June 1975
UTUCDS-R-75-731

AQV Analytical (AQV)
User's Guide and Program Description

by

James Larson
R. S. Michalski

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

June 1975

This work was supported in part by the National Science Foundation
under Grant No. DCR 74-03514.
AQVAL/1 (AQ7)  
User's Guide and Program Description

ABSTRACT

AQVAL/1 (AQ7) is a PL/1 program which synthesizes quasi-optimal formulas of the variable-valued logic system $VL_1$. By 'quasi-optimal formulas' we mean here disjunctive simple $VL_1$ formulas, which are optimal or sub-optimal with regard to a user-specified optimality functional.

The basic application of the program is in the area of machine learning and inductive inference ('inductive learning'): from descriptions of objects with known class membership, the program infers optimal or sub-optimal descriptions of object classes. These descriptions are expressed as $VL_1$ formulas and represent certain generalizations of inputted information. The program can also be used (at the appropriate setting of its parameters) for an efficient minimization of binary- or multi-valued switching functions with a large number of variables (e.g., 50-100 variables).
ACKNOWLEDGMENTS

Acknowledgments go to all people who contributed to different stages of the development of the current AQVAL/1 (AQ7) program and its earlier versions, in particular to Val Tareski, Peter Raulefs, Colso Frazer, Kurt Kirchert and Richard Chilauisky.

The authors would also like to acknowledge the National Science Foundation who partially supported this work under Grant No. DCR 74-03514.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. INPUT PARAMETERS AND FORMAT</td>
<td>2</td>
</tr>
<tr>
<td>2.1 Description of input parameters</td>
<td>2</td>
</tr>
<tr>
<td>2.2 Steps in the input specification</td>
<td>3</td>
</tr>
<tr>
<td>3. OUTPUT PARAMETERS</td>
<td>15</td>
</tr>
<tr>
<td>4. EXAMPLES OF PROGRAM INPUT AND OUTPUT</td>
<td>16</td>
</tr>
<tr>
<td>5. PROGRAM IMPLEMENTATION</td>
<td>39</td>
</tr>
<tr>
<td>5.1 General remarks</td>
<td>39</td>
</tr>
<tr>
<td>5.2 Program representation of complexes and events</td>
<td>39</td>
</tr>
<tr>
<td>5.3 Internal structure for storing intermediate stars</td>
<td>40</td>
</tr>
<tr>
<td>5.4 Structure of the program</td>
<td>42</td>
</tr>
<tr>
<td>5.5 Basic procedures in the program</td>
<td>43</td>
</tr>
<tr>
<td>5.6 Remarks about other related programs</td>
<td>46</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>48</td>
</tr>
<tr>
<td>APPENDIX A</td>
<td>50</td>
</tr>
<tr>
<td>APPENDIX B</td>
<td>51</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

This paper provides a user's guide and brief description of the program* AQVAL/1 version AQ7. The program infers an optimal or sub-optimal (with regard to a user-specified optimality functional) disjunctive simple formula(s) of the variable-valued logic system VL₁. In this version, ** the formulas are inferred from event sets (i.e., sets of sequences of values of input variables) where each set is associated with a certain value of the output variable(s). The way in which sets are associated with values of output variable(s) depends on the mode of program operation, as explained in section 2.2. in the description of parameter MODE.

It is assumed here that the reader is acquainted with the definition of the variable-valued logic system VL₁ given in paper¹ and therefore very limited explanation of the concepts of VL₁ is included.

Chapters 2 and 3 describe in detail the input and output parameters of the program. Chapter 4 gives two complete examples of actual specification of these parameters. Chapter 5 describes briefly the implementation of the synthesis algorithms (described in paper²). In order to understand the last chapter, the reader should be acquainted with paper². Bibliography, in addition to papers containing the background information for this paper (i.e., the definition¹ of the VL₁ and a description of synthesis algorithms²) includes also the complete list of papers relevant to the current status of variable-valued logic and its

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

** A synthesis of quasi-optimal formulas from other (non-optimal) formulas (rather than from event sets) is presently implemented only in version AQ9 of AQVAL/1 described in paper²⁰.
applications. In Appendix A a short form describing steps in input preparation is given together with a list of optimality criteria.

The described here version AQ7 is written in PL/1 language* and consists of approximately 860 statements. The object module requires about 60k bytes (together with the PL/1 library routines about 90k) plus storage for the data, which may take somewhere between 10-150k bytes, depending on the total number of events, number of variables, sizes of the domains of variables, and some other parameters (minimum ≈112k bytes). The complete listing of the program is given in Appendix B.

2. INPUT PARAMETERS AND FORMAT

2.1 Description of input parameters

Following is a description of all input parameters in the order and format in which they should be specified in the input stream to the program. The description of each parameter consists of:

i. the name of the parameter,
ii. an example of its specification
iii. a list of possible values of the parameter
iv. a description of its meaning.

Input parameters are divided into 2 groups:

A. Control parameters (specified in the PL/1 input data format) which supply an information about the way the program should be executed.

Specification of these parameters ends with a semi-colon (;).

If any control parameter is not specified by the user in his input data then it takes a default value. The default value is indicated as the value used in the example of the parameter specification.

* It is adapted for optimizing compiler and runs under the HASP operating system.
B. Data parameters which describe the VL function(s) to be optimized and optimization functional.

2.2 Steps in the input specification:

I. Specification of control parameters:

- MODE

Example: \( \text{MODE} = 'IC' \)

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

This parameter defines the mode of program operation that is, it defines how the program should interpret the data parameters (specifically, event sets). Consequently, the interpretation of the output from the program will also be dependent on this parameter.

Here we will present a brief description of each mode (IC, DC, VL), and in Chapter 4 we will illustrate each mode by an example. (For a more detailed description of these concepts, consult paper 2.)

For the convenience of the description of MODE parameter, let us assume that in the input stream there are \( m \) (disjoint) event sets (step IV, p. 11, parameter P) which we will denote here as \( P^0, P^1, \ldots, P^{m-1} \). (The assignment of symbols \( P^0, P^1, \ldots, P^{m-1} \) to the event sets specified in the input stream depends on the CLIST parameter (step IV, p. 11)).

-- If MODE = 'IC' ('Intersecting Covers'), then the event sets are treated as defining a set of \( m \) binary-valued VL functions \( f_i \), \( i = 0, 1, \ldots, m-1 \). A function \( f_i \) takes value 1 for events from \( P^i \), and value 0 for the remaining events, specified in \( P^j \), \( j \neq i \), \( j = 0, 1, \ldots, m-1 \):

\[
f_i(e) = \begin{cases} 
1, & \text{if } e \in P^i \\
0, & \text{if } e \in P \setminus P^i 
\end{cases}
\]

where \( P \) is the union of all events specified in the program, i.e.

\[ P = P^0 \cup P^1 \cup \ldots \cup P^{m-1} \]

The output from the program consists of \( m \) DVL formulas which are optimized expressions \( V(f_i) \) (with regard to the optimality functional \( A \), step IV, p. 11, parameter NCRIT) CLIST(NCRIT) TLIST(NCRIT) of functions \( f_i \).
These expressions generalize the input information about functions in the sense that they may now assign a specific value (1 or 0) to events which were not listed in the input stream (i.e., events from $E \setminus F$, called *-events, where $E$ is the universe of events). Formally, a $V(f_i)$ expression is defined as a DVL1 expression corresponding to an optimized (with regard to functional $A$) cover of set $F$ against the set $F\setminus F^i$:

$$V(f_i) \leftrightarrow C_i = C^{(i)} \mid D^{(i)} \quad i=0,1,\ldots,m-1$$

In this mode some *-events can satisfy more than one formula $V(f_i)$, i.e., the intersection of some formulas $V(f_i)$ may not be empty (hence the mode is called 'Intersecting Covers'). Note, however, that each event from $F$ (i.e., event from the input list) satisfies exactly one formula (it follows from the assumption that sets $F$ are disjoint).

-- If MODE = 'DC' ('Disjoint Covers'), then the input event sets are treated similarly as in the case when MODE = 'IC'. The difference is, however, that the program will produce formulas $V(f_i)$ which are now all disjoint (a given *-event will satisfy at most one output formula $V(f_i)$). Formulas $V(f_i)$, $i=0,1,\ldots,m-1$, obtained in this mode, correspond to consecutively synthesized covers $C_i$:

$$V(f_0) \leftrightarrow C_0 = C^{(0)} \mid \bigcup_{j=1}^{m-1} F^j$$

$$V(f_1) \leftrightarrow C_1 = C^{(1)} \mid C_0^U \bigcup_{j=2}^{m-1} F^j$$

$$\vdots$$

$$V(f_i) \leftrightarrow C_i = C^{(i)} \mid \bigcup_{j=0}^{i-1} C_0^U \bigcup_{j=i+1}^{m-1} F^j$$

$$\vdots$$

$$V(f_{m-1}) \leftrightarrow C_{m-1} = C^{(m-1)} \mid \bigcup_{j=0}^{m-2} C_0^U$$
(By $c^0_j$ is meant the union of complexes in cover $C_j$.)

Each formula $V(f_i)$ is a union of terms, each term corresponding to a complex in the cover $C_i$.

As we see, the formulas depend on the assumed order of sets $F_i$ (see parameter OLIST in step IV p. 11).

-- If MODE = 'VL', then sets $F_i$, $i=0,1,2,\ldots,m-1$ are treated as sets defining one $m$-valued VL function:

$$f: \mathbf{E} \rightarrow \{0,1,\ldots,m-1\}$$

where

$\mathbf{E}$ -- the event space defined as $D_1 \times D_2 \times \ldots \times D_n$.

($D_i$ - domains of individual variables)

Sets $F_i$ are related to the function as follows:

$$F_i = \{ e \in \mathbf{E} | f(e) = i \} \quad i = 0,1,\ldots,m-1$$

The output from the program is an optimized $DL_k$ formula $V(f)$ expressing function $f$. Formula $V(f)$ is defined as:

$$V(f) = (m-1)(T_1^{m-1} \lor T_2^{m-1} \lor \ldots) \lor$$

$$(m-2)(T_1^{m-2} \lor T_2^{m-2} \lor \ldots) \lor$$

$$\vdots$$

$$\lor$$

$$1 \quad (T_1^1 \lor T_2^1 \lor \ldots)$$

where

$T_i^{m-1}$, $i=1,2,\ldots$ -- terms corresponding to complexes in the quasi-optimal (q-o) cover

$$C_{m-1} = C(F^{m-1}_{i=0} | F_1)$$

$T_i^{m-2}$, $i=1,2,\ldots$ -- terms corresponding to complexes in the q-o cover

$$C_{m-2} = C(F^{m-2}_{i=0} | F_1)$$

$$\vdots$$

$T_i^{1}$, $i=1,2,\ldots$ -- terms corresponding to complexes in the q-o cover

$$C_1 = C(F^1_{|F^0})$$
Note that the total number of covers \( C_1 \) is smaller by 1 than the number of sets \( P_1 \).

**INFORM**

Example: \[ \text{INFORM} = '\text{VECTOR}' \]

Possible values: 'VECTOR', 'GAMMA', 'VL' *

If \text{INFORM} equals 'VECTOR', then events are read in as sequences of variable values. If \text{INFORM} equals 'GAMMA', then events are read in the form of event numbers \( \gamma(e) \). If \text{INFORM} = 'VL', then events are expressed as a VL formula.

**TITLE**

Example: \[ \text{TITLE} = 3 \]

Possible values: any non-negative integer

This parameter specifies the number of lines reserved for the problem title (which is typed just after the semi-colon which ends the specification of control parameters).

**MAXSTAR, CUTSTAR**

Example: \[ \begin{align*}
\text{MAXSTAR} &= 150 \\
\text{CUTSTAR} &= 50
\end{align*} \]

Possible values: \( \text{MAXSTAR} \) any positive integer smaller than \( \text{NGB} \) value (see next parameter)

\( \text{CUTSTAR} \) any positive integer smaller than \( \text{NGB} \) value (see next parameter)

These parameters are used to control the speed of the program. Small values of these parameters (e.g., \( \text{MAXSTAR} = 5 \), \( \text{CUTSTAR} = 3 \)) will cause the program to run more quickly (but may also cause the program to generate less optimal formula(s)).

The work of these parameters can briefly be described as follows: A star generation is a multi-step process which at each step produces a set of complexes ('intermediate star'). If at any step an intermediate star contains more than \( \text{MAXSTAR} \) complexes, then it is reduced to \( \text{CUTSTAR} \) complexes, selected as the 'best' according to the assumed optimality functional (see CLIST and TLIST parameters).

*Implemented presently only in version AG described in paper 20.*
• **NGE**

Example: **NGE = 200**

Possible values: any positive integer (smaller than $2^{15}$)

This parameter denotes the maximal number of complexes which can be stored during a start generation. **NGE** defines the maximum size of a list $G$ each element of which consists of 4 fields:

1. storing a complex (which is a bit string of length $\sum_{i=1}^{n} d_i$)
2. storing a value (real number) of a criterion under consideration for the complex
3. and 4. pointer to the next and to the previous element of the list.

• **NMQ**

Example: **NMQ = 25**

Possible values: any positive integers

This parameter defines the length of a list reserved for storing complexes $L^q$ ('quasi-extremals' or 'best complexes') constituting the final formula(s).

• **LQST**

Example: **LQST = '1'B**

Possible values: 'O'B, '1'B

If LQST = '1'B each complex $L^q$ selected from a start will be 'reduced' before it is put on the output list. The reduction means that the references (ranges) of variables in each complex will be reduced as much as possible, providing that

1. the complex will still cover the same number of events in the original set $F_i$
2. the reduced complex will have the same number of literals (i.e., selectors in the corresponding term is the formula).

*See page 8 of the program listing in Appendix (separate report).*
SAVE

Example:  SAVE = '0'B
Possible values: '0'B, '1'B

If SAVE = '0'B, the output formula is not stored.
If SAVE = '1'B, then the formula is stored in the disc file
called 'COVER' (the disc file is specified by JCL cards
listed with example 1 at the end of this paper).

Individual complexes are stored as bit strings
('binary-positional' notation) in PL/I list format (see Chapter 5),
similarly as they are stored during the program execution.
This way of storing facilitates the evaluation of the formula.

QLQT

Example:  QLQT = '0'B
Possible values: '0'B, '1'B

This parameter controls the printing of summary
information about star generation. If QLQT equals
'0'B, no information will be printed. If QLQT equals
'1'B, then the program will print out a summary about each
star, the quasi-extremal, and the number of new events covered
by it in a given P*.

LQTRACE

Example:  LQTRACE = '0'B
Possible values: '0'B, '1'B

This parameter controls whether detailed information
about each quasi-extremal should be printed. If
LQTRACE = '0'B, then no information is printed. If
LQTRACE = '1'B, then in addition to the information
printed by QLQT, the actual cost of the quasi-extremal
and the array denoting the elements not covered by the
current and previous stars is printed.

STRACE

Example:  STRACE = '0'B
Possible values: '0'B, '1'B

This parameter controls whether detailed information
about star generation should be printed. If STRACE = '0'B,
then detailed information is not printed. If STRACE = '1'B,
each step of star generation is printed.
QST

Example: \[ \texttt{QST} = '0'B \]
Possible values: '0'B, '1'B

This parameter controls whether a summary of the steps in star generation should be printed. If QST = '0'B, then the summary is not printed. If QST = '1'B, then the summary is printed.

II. Separation between control and data parameters:

Type semi-colon (;) after the last control parameter. If no control parameters are specified (i.e., their default values are accepted), the semi-colon still should be typed.

III. Specification of the problem title:

Type the title of the program using as many lines as were specified by TITLE control parameter. The title information will be reproduced verbatim at the top of the output page.

IV. Specification of data parameters

All of the following data parameters are in PL/1 LIST format; that is, a value must be present for each of these parameters in the order specified. There are no default values for these parameters.

- **NSPEC TYPE TYPELIST(NSPEC)**

Example: \[ \texttt{2 'FACTOR' 1 4} \]
Possible values: NSPEC -- a positive integer \( \leq 64 \) (this restriction can be easily removed)
TYPE -- 'FACTOR' or 'INTERVAL'
TYPELIST -- a list of NSPEC positive integers \( \leq 64 \).

These parameters specify the type of each variable which is used in the event description. NSPEC specifies the number of variables which will be of the type given by the parameter TYPE. The parameter TYPE specifies the type of the variables whose indices are specified by TYPELIST. (All variables not in TYPELIST are assumed to have the 'opposite' type.) There should be NSPEC indices in TYPELIST. In the example above, there are NSPEC = 2 variables which are given the type
'FACTOR' (i.e., x1 and x4). The rest are given the 'opposite' type namely 'INTERVAL' (i.e., x2, x3, x5, ...). Note that variable indices begin with 1.

The following are some guidelines for selecting the type of each variable.

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

It should be noted that interval variables are simpler for evaluation than factor variables. Also, the synthesis of optimal \( \mathcal{F}_n \) formulas usually will take considerably less computational time and memory if variables are specified as interval rather than as factor variables. On the other hand, \( \mathcal{F}_n \) formulas with interval variables will usually have considerably more terms and selectors than formulas with factor variables (when no restrictions on the variables are assumed).

- **NV NLEV**(NV)

  **Example:**
  
  \[4 \quad 5, \quad 3, \quad 2, \quad 4\]

  Possible values: NV -- any positive integer \( \leq 64 \) (this restriction can be easily removed)  

  NLEV -- a list of NV positive integers

  These parameters specify the number of variables (NV) and the number of levels NLEV(i) for each variable xi which is used in the description of the input events. Since the smallest variable value allowed is 0, NLEV(i) (the number of levels for the \( i \)th variable) should be one larger than the largest possible value of the \( i \)th variable. In the example above, there are 4 variables: variable x1 has 5 possible values (i.e., the domain of x1 consists of 0,1,2,3, and 4); x2, x3, and x4 have 3, 2 and 4 possible values, respectively.

- **NCL NE**(NCL)

  **Example:**
  
  \[3 \quad 5, \quad 1, \quad 2\]

  Possible values: NCL -- any positive integer  

  NE -- a list of NCL positive integers
These parameters specify the number of event sets (NCL) which are in the input stream and the number of events for each set (NE). In the above example, there are 3 event sets, the first has 5 events, the second has only 1 event, and the third has 2 events.

- **OLIST (NCL)**

  Example: [1, 0, 2]

  Possible values: some ordering of the integers 0,1,...,NCL-1 (NCL as specified above)

  This parameter specifies the way in which the input event sets are assigned symbols F_i, i=0,1,...,NCL-1. The consecutive events sets in the input stream are numbered 0,1,2,...,NCL-1. The first number in the OLIST tells which event set in the input stream is assigned symbol F_{NCL-1}, the ith number tells which event set is assigned symbol F_{NCL-i}. In the example, the second (numbered as 1) event set is assigned symbol F_2, the first (numbered as 0) is assigned symbol F_1, and the third (numbered as 2) is assigned F_0.

- **NCRIT CLIST TLIST**

  Example: [3, 1, 2, 4, 0, 0, 0, 1]

  Possible values: NCRIT -- a positive integer between 1 and 7
                  CLIST -- a list of NCRIT positive integers between 1 and 7
                  TLIST -- a list of NCRIT real values from the interval [0,1]

  These parameters specify the optimality functional 2,12 which is to be used in finding the 'best' complex 1,12. NCRIT is the number of cost criteria which are used. CLIST specifies criteria to be used by listing the criteria numbers in order of criteria preference. The criteria numbers are numbers assigned to each criterion as described below. TLIST specifies the tolerance which is to be allowed for each respective cost criterion.

  The cost functional is used to select the 'best' complex from a star (a set of complexes which are maximal under inclusion and which cover the same event of F_1 but no events of F_0). The process proceeds as follows: each criterion is selected in turn beginning with CLIST(1). Using the selected criterion, a value is assigned to each complex.
in the star. The minimum (MINVAL) and the maximum (MAXVAL) cost of each complex in the star is computed. Using the associated tolerance limit TLIST(i), a threshold UBOUND is calculated.

\[ UBOUND = MINVAL + TLIST(i) \times (MAXVAL - MINVAL) \]

All complexes whose cost is greater than UBOUND are eliminated from further consideration. If this leaves only one complex, the process terminates and this is the required 'best' complex. Otherwise, the next specified criterion is selected and the process is repeated. If all specified criteria have been used and more than one complex remains, a complex with minimum cost with respect to the last criterion is selected as the 'best' complex (i.e., quasi-extremal).

The user specified criteria characterize the formula which he wants to obtain. Since the program builds a formula from selected complexes, instead of 'user criteria' (referring to the whole formula) the program uses 'program criteria' which refer to the individual complexes. These program criteria are designed to best approximate the original criteria (which refer to the whole formula). For example, if the user specifies 'minimum number of terms' as a criterion, then this criterion is substituted in the program by the criterion 'maximum number of events which a given complex covers in the remaining set of events to be covered'.

Below are described currently available user criteria and corresponding to them program criteria (in parentheses):

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Notation</th>
<th>Description of Attributes (Criteria)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>t --</td>
<td>the number of terms in a formula</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(The negative of the number of events in the event set which are covered by the given complex but not by any of the previously determined complexes (stored in (N[i])). The negative is used to have the minimum value for the criterion when the complex covers the maximum number of events.)</td>
</tr>
<tr>
<td>2</td>
<td>s --</td>
<td>the number of selectors in a formula</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(The number of selectors in a complex.)</td>
</tr>
<tr>
<td>3</td>
<td>z --</td>
<td>sum of the costs of the variables in a formula (z(v))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(The sum of the costs of the variables in a complex: [z' = \sum_{i \in I} z(i)])</td>
</tr>
</tbody>
</table>
\[ z(i) \] -- cost of the variable \( x_i \) which is specified by the \( Z \) parameter (see \( Z \) parameter specification below)

\( I \) -- set of indices of variables which appear in the complex.)

4

\[ g \] -- The degree of generalization
(The degree of generalization of a complex:
\[ \frac{c(E)}{c(E \cap F)} \]
\( c(E) \) -- number of events in the given complex \( E \)
\( c(E \cap F) \) -- number of events covered by the complex \( E \) which are members of the event set \( F \) which is being covered.)

5

\[ W \] -- 'Weight distribution'
The negative of the sum of weights of events of the event set which are covered by this complex but not by any previous complex. If this criterion is selected, weights \( W(K, J) \) are specified (see \( W \) parameter specification below). If the complex is created to cover event set \( K \), then the cost is given by
\[ - \sum_{J \in E} W(K, J) \]

\( E \) -- set of events of the event set \( k \) which are covered by this complex and not by any previous complex.)

6

\[ l \] -- Total length of references
(The length of references in a complex. This is the sum of the number of constants which appear in the reference of each of the selectors of the complex (in expanded form)
\[ \sum_{i \in I} l(x_i) \]

\( I \) -- as in 3 above
\( l(x_i) \) -- number of constants in the reference of the selector for the variable \( x_i \).)

7

\[ r \] -- The relative scope of references
(The relative scope of references in a complex. This is the sum of the mean deviations for variables in the complex. The mean deviation of a variable is the average absolute difference between the reference constants and the mean of these constants. That is,
\[ \text{cost} = \sum_{i \in I} (\text{MD}_i) \]
\[ MD_i = \sum_{r \in R_i} \frac{|\text{avg}_i - r|}{c(R_i)} \]

I -- set of indices of variables which appear in the complex

avg\(_i\) -- average value of the constants in the reference for variable \(x_i\) in the complex

\(R_i\) -- the set of constants \(r\) in the reference of the variable \(x_i\) in the complex

c(\(R_i\)) -- the number of elements of \(R_i\)

- \(F\)

Example: 0 0 1 0 2 0 1 3 2

1 0 5 1 1 5 2 0 2

Possible values: a set of nonnegative integers which properly represent each event

\(F\) is a list of event sets \(F_i\) defining the given VL function(s). \(F\) must be completely specified (no default values). If INFORM = 'VECTOR', then each event must be represented as a set of NV (number of variables) integers each of which specifies the respective variable value for that event. The value for variable \(x_i\) must be less than NLEV(\(i\)). If INFORM = 'GAMMA', then the gamma representation for each event must be given.

The example above could be the representation for 6 events in 'VECTOR' format with NV = 3.

The following parameters (\(Z\) or \(W\)) are specified only if the cost criterion 3 or 5 are specified, respectively. They are specified in PL/I data format and each specification is terminated with a semi-colon (;

- \(Z\)

Example: \(Z(1) = 2\) \(Z(4) = 1.2;\)

Possible values: any real numbers

This parameter is specified only if criterion 3 is specified and specifies the cost of each variable (i.e., the cost of \(x_i\) is specified by \(Z(1)\)). Any variables for which no specification is given are assigned cost = 1. In the example given, variable \(x_1\) has cost 2, variable \(x_4\) has cost 1.2, and the other variables have cost = 1.
\begin{itemize}
  \item \textbf{W}
  \begin{quote}
  Example: \( W(0,2) = 1.5 \quad W(1,1) = 2; \)
  \end{quote}
  Possible values: any real numbers

  This parameter is specified only if criterion 5 is specified. \( W(K,F) \) specifies the weight of \( J^{th} \) \((J=1,2,3,\ldots)\) event in the \((K+1)^{st} \) \((K=0,1,\ldots)\) event set in the list \( F \). Any event for which no value is given is assigned weight = 1. In the example above, the second event of the first event set is assigned weight = 1.5, the first event of the second event set is given weight = 2.0, the rest of the events are given weight = 1.0. (Note event set numbering begins at 0, event numbering within each event set begins with 1.)

  This concludes the parameters specification for one problem. Several problems may be run together by specifying all of the parameters for each problem in succession. No parameter values (except the default control parameter values) are carried over from one problem to the next.

\end{itemize}

3. OUTPUT PARAMETERS

Some of the output parameters of the program are reflections of the input parameters. The following parameters are generated by the program (the text enclosed in the dotted line is the actual information printed on the printout from the program):

\begin{itemize}
  \item Unused core
  \begin{quote}
  Example: \[ \text{AMOUNT OF UNUSED CORE} = \text{4k} \]
  \end{quote}
  This parameter specifies the amount of core memory which was estimated but not used.

  \item COMPLEX:
  \begin{quote}
  Example: \[ \text{COMPLEX: (x1 = 0 2) (x3 = 1) (9, 6, 2, 12)} \]
  \end{quote}
  This parameter specifies the actual complexes which were synthesized. Complexes of one formula (or in the case of "Y" mode) complexes which have the same coefficient are grouped together under the heading identifying the event
class which they cover. Together with each complex is printed a list of 4 numbers (COV, NEW, IND, TOT).

where:

COV -- denotes the number of events covered by the complex in the given event set $P_i$ (i.e., event set associated with the current decision class)

NEW -- is equal to COV minus number of events in $P_i$ covered by the previous complexes on the output list ('new covered').

IND -- is equal to COV minus number of events in $P_i$ covered by all the other complexes in the cover ('independently covered')

TOT -- is equal to the total number of events in $P_i$ which are covered by this complex and previous complexes in the cover

• DELTA

Example: [DELTA for this set is C]

This parameter specifies an estimate (upper bound) on the distance (measured in number of complexes) of the formula produced from the minimal formula. In the above example, since DELTA is 0 the set of complexes is minimal. Note that if stars are trimmed (i.e., an intermediate star exceeds MAXSTAR complexes) then DELTA is no longer a valid estimate. If complexes are a result of trimmed star, a message to this effect is printed.

• INTERMEDIATE STAR SIZE

Example: [Largest intermediate star size is 56]

This parameter specifies an estimate of the maximum number of complexes which are stored in the star list. The NSE parameter may be reduced to approximately MAXSTAR plus this estimate if this problem is run again.

4. EXAMPLES OF PROGRAM INPUT AND OUTPUT

Two examples of the input and output formats for the program AQUAL/1 (AG7) are presented below. The first example is further illustrated with GLD representations of the formulas produced by the program. Included with each line of the input examples is a description which lists the names
of the parameters which are specified on that line, a brief indication
of the significance of each parameter and the actual values which the
parameter assumes in the example.

The first few lines of each example of program input is the
JCL specification. These cards must be included before any parameter
specification to gain access to the program and distribute the program
output properly. Preceding the JCL specification must be the ID cards
which contain the time and region estimates. The amount of these
resources which the program requires is a complex function of the
control parameters and the dimension and size of the event sets which
are to be covered. Samples of various parameters and the amount of time and
region to successfully complete the execution of the program are given in
Figure 1. In general, for small problems, the region estimate must be
at least 116 k and the time estimate can be less than 30 seconds. For
very large problems, a region estimate of 180 k and time of 6 minutes
is probably sufficient. Note that the region estimate must appear on both
the ID card and the EXEC card of the JCL. Additional blank lines may be
inserted for clarity in several points in the input. They must not appear,
however, between any JCL cards or between the semi-colon which terminates
the control parameters and the title (unless the control parameter TITLE
has allowed for this).

The output of the program begins with the title which was
specified in the input. The values of all control parameters, the type
of each variable and event set sizes are listed. Then, the amount of
core remaining is given, which is the amount of core which was estimated
but not used. Next, the OLIST and optimization criteria are listed
followed by the actual event sets which were specified. The VL formula(s)
follow. Each complex (or term) in the synthesized formula(s) is printed along the DELTA for this set of complexes. DELTA gives an estimate (upper bound) of the number of complexes by which this formula exceeds the minimal formula. If the formulas are a result of trimmed stars (the size of some intermediate star exceeded the MAXSTAR parameter), a message is printed which states that the complexes are a result of trimmed stars and thus the value of DELTA is only an estimate (not an upper bound) of the distance of the formula generated to the minimal formula. The largest star size and the largest intermediate star size are printed. The largest intermediate star size plus the value of MAXSTAR gives an estimate of the minimum size of the parameter NGE which was actually necessary in the synthesis of this set of complexes. After the entire set of formula(s) has been printed, a message indicating a normal termination is given which indicates the successful completion of the synthesis of formulas for this problem.

The first example demonstrates the three modes of operation of the program as specified by the parameter MODE. In addition to the input stream and the output generated, the formulas which were generated are illustrated in Fig. 2, 3 and 4 by Generalized Diagrams (GLD). The second example demonstrates the use of criteria 3 and 5, the specification of the associated cost and weight parameters (Z and W), and the different tolerance specifications for each criterion.

GLD Description for Example #1

The GLD is a graphical model of the event space and the complexes which cover the events in this space. Each square on the GLD represents one event in the event space; a set of events enclosed in an
area or set of areas can represent a complex. In this GLD, the
variables $x_1$ and $x_2$ specify the row in which a particular event is to
be placed, the variables $x_3$ and $x_4$ specify the column. An event set
is a set of squares with the same number. If the event sets are to be
covered in VL mode, then the numbers (1) in the squares correspond to
the set $F^1$ and to the terms of the $2VL_1$ formula with coefficient 1. If the
event sets are to be covered in IC or DC mode, then the numbers (1)
correspond to those events with a function value of 1 for the $i^{th}$
function and a function value of 0 for the other functions. The events
which are given to the program are indicated in the diagrams by numbers
in the squares. The complexes of each type of cover ($VL, IC, DC$) which were
generated by the program are illustrated in the respective diagrams
(Fig. 2, 3, 4) by open or shaded areas together with their $VL_1$
representation. Some observations about these three different covers
follow.

**VL mode:** The complex which covers the event set $F^1$, does not
cover events of any of the sets $F^i$, $i=3,2,1,0$. The complex which covers the
event set $F^2$, however, also covers some events of sets $F^3$ and $F^0$. Since this
formula is evaluated by testing the complexes associated with event sets
$F^4$ and $F^3$ before the complex associated with the set $F^2$, the events in
the original event sets are classified correctly.

**IC mode:** Each complex does not cover any events from any event sets
other than the one which it was specified to cover. The complexes which
cover event sets $F^2$ and $F^3$ intersect with the complex which covers event
set $F^0$ in an unspecified area of the event space. It is thus possible
that an event in this area could satisfy two different formulas. (Such
a situation could be interpreted as an uncertain classification.)
DC mode: The complexes are similar to those produced in IC mode. However, they do not intersect in any area of the space hence at most one formula is satisfied by the set of complexes for any event in the event space.
Estimates of Upper Bounds for Region and Time Requirements in Various Sizes of Problem (on IBM 360/75).

<table>
<thead>
<tr>
<th>No. of Variables</th>
<th>Total No. of Levels</th>
<th>No. of Classes</th>
<th>Total No. of Events</th>
<th>MIN</th>
<th>MODE</th>
<th>MAXSTAR</th>
<th>CUTSTAR</th>
<th>Region</th>
<th>Time (min, sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>15</td>
<td>5</td>
<td>23</td>
<td>200</td>
<td>VL</td>
<td>IC, DC</td>
<td>100</td>
<td>90</td>
<td>114k (0,6)</td>
</tr>
<tr>
<td>16</td>
<td>72</td>
<td>2</td>
<td>42</td>
<td>200</td>
<td>VL</td>
<td>15</td>
<td>5</td>
<td>132k (0,15)</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>162</td>
<td>2</td>
<td>78</td>
<td>500</td>
<td>VL</td>
<td>5</td>
<td>5</td>
<td>160k (1,6)</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>109</td>
<td>2</td>
<td>176</td>
<td>500</td>
<td>VL</td>
<td>3</td>
<td>1</td>
<td>156k (2,34)</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>130</td>
<td>3</td>
<td>290</td>
<td>280</td>
<td>IC</td>
<td>6</td>
<td>3</td>
<td>180k (40,0)</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>132</td>
<td>15</td>
<td>300</td>
<td>280</td>
<td>IC</td>
<td>7</td>
<td>4</td>
<td>180k (18,0)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1
Graphical Representation of the $VL_1$ Formula Obtained
Graphical Representation of the Set of $\mathcal{VL}_1$ Formulas
Obtained in Example 1 (IC mode).

Figure 5
Graphical Representation of the Set of $VI_1$ Formulas Obtained in Example 1 (DC mode).
Input Description

1. JCL
2. Control parameters (those not specified assume default values)
3. Title (3 lines of the title)
4. INSPEC TYPE Typelist: variable type specification (x2 is a factor variable, the rest are type intervals)
5. NV: number of variables (variables)
6. NLEV: number of levels for each variable (variable x2 has 3 levels, the rest have 4 levels)
7. NCL NE: event set description (there are 5 event sets, the first 2 have 4 events, the rest have 5 events)
8. OLIST: ordering of event sets (the sets are to be covered in the order of input)
9. NCRIT CLIST TLIST: cost functional description (2 criteria are to be used, #1 first, then #2 each with 0 tolerance)
10. Event sets

Input Data for Example #1
Figure 5
### Example #1: MODE = 'IC' (Control parameters)

```
MODEL=IC
TITLE=3
NIE=100 MAXSTAR=100 CUTSTAR=90;
```

#### Example data

<table>
<thead>
<tr>
<th>Factors 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
<tr>
<td>4.3.4.4</td>
</tr>
<tr>
<td>54555</td>
</tr>
<tr>
<td>01234</td>
</tr>
<tr>
<td>21200</td>
</tr>
</tbody>
</table>

#### Data parameters

| 0000       |
| 0202       |
| 0103       |
| 1101       |
| 0112       |
| 1011       |
| 1113       |
| 1210       |
| 2102       |
| 3100       |
| 2123       |
| 3120       |
| 3123       |
| 2012       |
| 2210       |
| 2023       |
| 2220       |
| 2223       |
| 0030       |
| 0231       |
| 0233       |
| 1033       |
| 1230       |

### Example #1: MODE = 'DC' (Control parameters)

```
MODEL=DC
TITLE=3
NIE=100 MAXSTAR=100 CUTSTAR=90;
```

#### Example data

<table>
<thead>
<tr>
<th>Factors 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
<tr>
<td>4.3.4.4</td>
</tr>
<tr>
<td>54555</td>
</tr>
<tr>
<td>01234</td>
</tr>
<tr>
<td>21200</td>
</tr>
</tbody>
</table>

#### Data parameters

| 0000       |
| 0202       |
| 0103       |
| 1101       |
| 0112       |
| 1011       |
| 1113       |
| 1210       |
| 2102       |
| 3100       |
| 2123       |
| 3120       |
| 3123       |
| 2012       |
| 2210       |
| 2023       |
| 2220       |
| 2223       |
| 0030       |
| 0231       |
| 0233       |
| 1033       |
| 1230       |
EXAMPLE #1  MODE OF OPERATION IS 'VL'

SPACE ALLOCATED FOR GIE# IS 100  SPACE ALLOCATED FOR MQ IS 25
MAXIMUM STAR SIZE BEFORE TRIMMING WILL BE CNK 100 THEN IT WILL BE CUT TO 90 MODE = VL
LQ TRACE = 0  STAR TRACE = 0  QUICK LQ TRACE = 0  QUICK STAR TRACE = 0  SAVE COVER DATA = 0  SADEVOL= 0

INPUT FORMAT IS VECTOR
THE FOLLOWING VARIABLES ARE COVERED BY FACTORS
THE FOLLOWING VARIABLES ARE COVERED BY INTERVALS

NUMBER OF VARIABLES = 4
NUMBER OF LEVELS FOR EACH VARIABLE: 4 3 4 4
NUMBER OF EVENTS SPECIFIED FOR EACH CLASS:

CLASS  #EVENTS
  0  4
  1  5
  2  5
  3  4
  4  5

** AMOUNT OF UNUSED CORE = 2K **

CLIST= 0 1 2 3 4
MCPIT= 2
CLIST= 1 1
TLIST= 0.00 0.00
LQ-STAR OPTION LOST= 118

CLASS F1 (0)
EVENT NO.  1= 0 0 0 0
EVENT NO.  2= 0 0 0 0
EVENT NO.  3= 0 0 0 0
EVENT NO.  4= 0 0 0 0

CLASS F1 (1)
EVENT NO.  1= 0 1 1 0
EVENT NO.  2= 1 0 1 0
EVENT NO.  3= 1 0 1 0
EVENT NO.  4= 0 1 0 0

CLASS F1 (2)
EVENT NO.  1= 2 0 0 0
EVENT NO.  2= 0 0 0 0
EVENT NO.  3= 0 0 0 0
EVENT NO.  4= 0 0 0 0

Output for Example #1...
EVENT NO.  5  3  1  2  3
CLASS F(3)
EVENT NO.  1  2  0  1  2
EVENT NO.  2  2  2  1  0
EVENT NO.  3  2  0  2  3
EVENT NO.  4  2  2  2  0
EVENT NO.  5  2  2  2  3

CLASS F(4)
EVENT NO.  1  0  0  3  0
EVENT NO.  2  0  2  3  1
EVENT NO.  3  0  2  3  3
EVENT NO.  4  1  0  3  3
EVENT NO.  5  1  2  3  0

The following 1 Cartesian complexes cover the events in class 0:

Complex:  \((x_1=0,1) (x_3=0)\)

**** Delta for this set is 0 ****

The largest star had 7 elements; the largest intermediate star had 8 elements

The following 1 Cartesian complexes cover the events in class 1:

Complex:  \((x_1=0,1) (x_3=1)\)

**** Delta for this set is 0 ****

The largest star had 11 elements; the largest intermediate star had 11 elements

The following 1 Cartesian complexes cover the events in class 2:

Complex:  \((x_2=1)\)

**** Delta for this set is 0 ****

The largest star had 6 elements; the largest intermediate star had 8 elements
THE FOLLOWING 1 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS 3

COMPLEX: \( I \times 1 = 2 \) \( I \)

**** DELTA FOR THIS SET IS 0 ****

THE LARGEST STAR HAD 6 ELEMENTS; THE LARGEST INTERMEDIATE STAR HAD 6 ELEMENTS

*** NORMAL TERMINATION ***
EXAMPLE # 1 MODE = 'IC'

SPACE ALLOCATED FOR G1E IS 100
SPACE ALLOCATED FOR MQ IS 25

MAXIMUM STAR SIZE BEFORE TRIMMING WILL BE G1E = 100 THEN IT WILL BE CUT TO 90 MODE = IC
LO TRACE = 0 STAR TRACE = 0 QUICK LO TRACE = 0 QUICK STAR TRACE = 0 SAVE COVER DATA = 0 SAYLQ = 0

INPUT FORMAT IS VECTCR

THE FOLLOWING VARIABLES ARE COVERED BY FACTORS
THE FOLLOWING VARIABLES ARE COVERED BY INTERVALS 1 2 3 4.

NUMBER OF VARIABLES = 4
NUMBER OF LEVELS FOR EACH VARIABLE: 4 3 4 4

NUMBER OF EVENTS SPECIFIED FOR EACH CLASS:

<table>
<thead>
<tr>
<th>CLASS</th>
<th>#EVENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

** AMOUNT OF UNUSED CORE = 2K **

CLIST= 0 1 2 3 4
NCRT= 2
CLIST= 1 2
TLIST= 0.00 0.00
LO-STAR OPTION LOST= '1' B

CLASS F1 01
EVENT NO. 1= 0 0 0 0
EVENT NO. 2= 0 2 0 2
EVENT NO. 3= 0 1 0 3
EVENT NO. 4= 1 1 0 1

CLASS F1 11
EVENT NO. 1= 0 1 1 1
EVENT NO. 2= 1 0 1 1
EVENT NO. 3= 1 1 1 3
EVENT NO. 4= 1 2 1 0

CLASS F1 21
EVENT NO. 1= 2 1 0 2
EVENT NO. 2= 3 1 0 0
EVENT NO. 3= 2 1 2 3
EVENT NO. 4= 3 1 2 0
EVENT NO. | 5 | 3 | 1 | 2 | 3
---|---|---|---|---|---
CLASS F1, 3:
EVENT NO. | 1 | 2 | 0 | 1 | 2
EVENT NO. | 2 | 2 | 2 | 1 | 0
EVENT NO. | 3 | 1 | 2 | 2 | 0
EVENT NO. | 4 | 2 | 2 | 2 | 0
EVENT NO. | 5 | 2 | 2 | 2 | 3

CLASS F1, 4:
EVENT NO. | 1 | 0 | 0 | 3 | 0
EVENT NO. | 2 | 0 | 2 | 3 | 1
EVENT NO. | 3 | 0 | 2 | 3 | 3
EVENT NO. | 4 | 1 | 0 | 3 | 3
EVENT NO. | 5 | 1 | 2 | 3 | 0

THE FOLLOWING 1 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS 0
COMPLEX: (X1= 0 1) (X3= 0 1)

**** DELTA FOR THIS SET IS 0 ****

THE LARGEST STAR HAD 7 ELEMENTS; THE LARGEST INTERMEDIATE STAR HAD 8 ELEMENTS

THE FOLLOWING 1 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS 1
COMPLEX: (X1= 0 1) (X3= 1 1)

**** DELTA FOR THIS SET IS 0 ****

THE LARGEST STAR HAD 14 ELEMENTS; THE LARGEST INTERMEDIATE STAR HAD 14 ELEMENTS

THE FOLLOWING 1 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS 2
COMPLEX: (X1= 2 3 1) (X2= 1 1)

**** DELTA FOR THIS SET IS 0 ****

THE LARGEST STAR HAD 7 ELEMENTS; THE LARGEST INTERMEDIATE STAR HAD 8 ELEMENTS
THE FOLLOWING 1 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS 3

COMPLEX: \((x_1= 2) \ (x_2= 0 \ 2)\)

**** DELTA FOR THIS SET IS 0 ****

THE LARGEST STAR HAD 11 ELEMENTS; THE LARGEST INTERMEDIATE STAR HAD 14 ELEMENTS

THE FOLLOWING 1 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS 4

COMPLEX: \((x_3= 3)\)

**** DELTA FOR THIS SET IS 0 ****

THE LARGEST STAR HAD 8 ELEMENTS; THE LARGEST INTERMEDIATE STAR HAD 9 ELEMENTS

*** NORMAL TERMINATION ***
SPACE ALLOCATED FOR GIEJ IS 100  SPACE ALLOCATED FOR HQ IS 25
MAXIMUM STAR SIZE BEFORE TRIMMING WILL BE DONE = 100 THEN IT WILL BE CUT TO 90 MODE = DC
LQ TRACE = 0  STAR TRACE = 0  QUICK LQ TRACE = 0  QUICK STAR TRACE = 0  SAVE COVER DATA = 0  SAVECLO = 0
INPUT FORMAT IS VECTOR
THE FOLLOWING VARIABLES ARE COVERED BY FACTORS
THE FOLLOWING VARIABLES ARE COVERED BY INTERVALS
NUMBER OF VARIABLES = 4
NUMBER OF LEVELS FOR EACH VARIABLE: 4 3 4 4
NUMBER OF EVENTS SPECIFIED FOR EACH CLASS:
CLASS EVENTS
 0 4
 1 4
 2 5
 3 5
 4 5
** AMOUNT OF UNUSED CORE = 2K **

CLIST= 0 1 2 3 4
NCRIT= 2
CLIST= 1 2
TLIST= 0.00 0.00
LQ-STAR OPTION LOST = '1'8
CLASS FI (0)
EVENT NO. 1= 0 0 0 0 0
EVENT NO. 2= 0 2 0 2 0
EVENT NO. 3= 0 1 0 3
EVENT NO. 4= 1 1 0 1
CLASS FI (1)
EVENT NO. 1= 0 1 1 2
EVENT NO. 2= 1 0 1 1
EVENT NO. 3= 1 1 1 3
EVENT NO. 4= 1 2 1 0
CLASS FI (2)
EVENT NO. 1= 2 1 0 2
EVENT NO. 2= 3 1 0 0
EVENT NO. 3= 2 1 2 3
EVENT NO. 4= 3 1 2 0
<table>
<thead>
<tr>
<th>EVENT NO.</th>
<th>5</th>
<th>3</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENT NO.</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>EVENT NO.</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>EVENT NO.</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>EVENT NO.</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>EVENT NO.</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

**LASS F1 (3)**

| EVENT NO. | 1 | 0 | 0 | 3 | 0 |
| EVENT NO. | 2 | 0 | 2 | 3 | 1 |
| EVENT NO. | 3 | 0 | 2 | 3 | 3 |
| EVENT NO. | 4 | 1 | 0 | 3 | 3 |
| EVENT NO. | 5 | 1 | 2 | 3 | 0 |

**COV NEW IND TOT**

| 4 | 4 | 4 | 4 |

---

The following 1 Cartesian Complexes cover the events in class 0:

Complex: 
\[(x1=0 \ 1) \ (x3=0)\]

**** Delta for this set is 0 ****

The largest star had 7 elements; the largest intermediate star had 7 elements.

The following 1 Cartesian Complexes cover the events in class 1:

Complex: 
\[(x1=0 \ 1) \ (x3=1)\]

**** Delta for this set is 0 ****

The largest star had 11 elements; the largest intermediate star had 11 elements.

The following 1 Cartesian Complexes cover the events in class 2:

Complex: 
\[(x1=2 \ 3) \ (x2=1)\]

**** Delta for this set is 0 ****

The largest star had 4 elements; the largest intermediate star had 4 elements.

<table>
<thead>
<tr>
<th>COV</th>
<th>NEW</th>
<th>IND</th>
<th>TOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
THE FOLLOWING 1 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS 3
COMPLEX: \((x_1 = 2) \quad (x_2 = 0) \quad (x_3 = 2)\)

**** DELTA FOR THIS SET IS 0 ****

THE LARGEST STAR HAD 1 ELEMENT; THE LARGEST INTERMEDIATE STAR HAD 2 ELEMENTS

THE FOLLOWING 1 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS 4
COMPLEX: \((x_1 = 0) \quad (x_2 = 1) \quad (x_3 = 3)\)

**** DELTA FOR THIS SET IS 0 ****

THE LARGEST STAR HAD 1 ELEMENT; THE LARGEST INTERMEDIATE STAR HAD 2 ELEMENTS

*** NORMAL TERMINATION ***
// EXEC PGM=GEPA07, REGION=116K
// STEPLIB DD DSN=USER.P2123.GEPA07, DISP=SHR
// SYSPRINT DD SYSOUT=A
// COVER DD SYSOUT=B, DSN=(RECFM=FB, LRECL=80, BLKSIZE=80, BUFNO=1)
// SYSIN DD *
NQ=25 TITLE=3:

1 'INTERVAL' 3
4 5 3 5 4
3 2 4 1
7 1 2 3 5 4 7 6
8 0 .1 1 .5 3 .7 5

4 2 4 3, 2 1 2 2, 1 1 1 1, 2 2 2 2, 2 1 2 1, 1 0 1 0
0 0 0
Z(2) = 5.0 Z(3) = 2.1
W(1,4) = 1.2 W(2,1) = 2.53 W(1,1) = 1.44

Input Data for Example #2

JCL

Control parameters (those not specified assume default values)
TITLE (3 lines of the title)

NSPEC TYPE TYPELIST: variable type specification (x3 is an interval variable, the rest are FACTOR)

MV NLEV: variable specification (there are 4 variables with the number of levels given by 5 3 5 4 for variables x1, x2, x3, x4, respectively)

NCL NE: event set description (there are 3 event sets: the first set has 2 events, the second has 4 events, the third has one)

QLIST: ordering of event sets (the second set is covered first, next the first set, and finally the third)

NCRIT QLIST: cost functional specification (all 7 cost criteria are to be used in the order given)

TLIST: tolerance specification for the cost functional: (each cost function has the tolerance given, criteria #1 has a tolerance of 0, criteria #6 has a tolerance of .75)

Event sets

Parameters for criterion #5: weights of variables (variables x2 and x3 have the assigned weights, the rest (i.e. x1) have weight 1)

Parameters for criterion #5: weights of events: (all weights are 1 except the first and fourth events of the second set have weights 1.44 and 1.2 respectively the event of the third set has weight 2.53)
/* EXEC PGM=GEPAQ7,REGION=116K */
/STEP18 DD DSN=USER.PE183.GEPAQ7,DISP=SHR
/SYSIN DD SYSOUT=A
/DSER DD SYSCOUT=*0,DCB=(RECFM=FB,LECL=88,BLKSIZE=88,BUFF=1)
/NORD=25 TITL=3:

********************************************************************************
EXAMPLE #
********************************************************************************

1 INTERVAL 3

4 5 3 5 4

3 2 4 1

1 2 0

7 1 2 3 5 4 7 6

0 0 1 1 .5 .3 .75

{ 4 2 4 3, 2 1 2 2 , 1 1 1 1, 2 2 2 2, 2 1 2 1, 1 0 1 0

3(3)=5.8

w(1,4)=1.2 w(2,1)=2.53 w(1,1)=1.44

Input Data for Example #2
Figure 7

JCL

Control parameters (those not specified assume default values)
TITLE (3 lines of the title)

INSPEC TYPE TYPELIST: variable type specification (x3 is an interval variable, the rest are FACTOR)

IV NLEV: variable specification (there are 4 variables with the number of levels given by 5 5 5 4 for variables x1, x2, x3, w4, respectively)

NCL NE: event set description (there are 3 event sets, the first set has 2 events, the second has 4 events, the third has one)

CLIST: ordering of event sets: (the second set is covered first, next the first set, and finally the third)

MKRT CLIST: cost functional specification (all 7 cost criteria are to be used in the order given)

TLIST: tolerance specification for the cost functional: (each cost function has the tolerance given, criteria #1 has a tolerance of 0, criteria #3 has a tolerance of .75)

Event sets

Parameters for criterion #3: weights of variables (variables x2 and x3 have the assigned weights, the rest (i.e. x1) have weight 1)

Parameters for criterion #5: weights of events: (all weights are 1 except the first and fourth events of the second set have weights 1.44 and 1.2 respectively, the event of the third set has weight 2.53)
SPACE ALLOCATED FOR GIEJ IS 200
SPACE ALLOCATED FOR MQ IS 25

MAXIMUM STAR SIZE BEFORE TRIMMING WILL BE DONE = 150 THEN IT WILL BE CUT TO 50 MODE= IC
LQ TRACE = 0 STAR TRACE = 0 QUICK LQ TRACE = 0 QUICK STAR TRACE = 0 SAVE COVER DATA = 0 SAVELQ= 0

INPUT FORMAT IS VECTOR

THE FOLLOWING VARIABLES ARE COVERED BY FACTORS
1 2 3

THE FOLLOWING VARIABLES ARE COVERED BY INTERVALS

NUMBER OF VARIABLES = 4

NUMBER OF LEVELS FOR EACH VARIABLE: 5 3 5 4

NUMBER OF EVENTS SPECIFIED FOR EACH CLASS:

<table>
<thead>
<tr>
<th>CLASS</th>
<th># EVENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 2</td>
</tr>
<tr>
<td>1</td>
<td>4 2</td>
</tr>
<tr>
<td>2</td>
<td>1 1</td>
</tr>
</tbody>
</table>

** AMOUNT OF UNUSED CORE = OK **

CLIST= 1 2 0

NCRT= 7

CLIST= 1 2 3 4 5 6 7

TLIST= 0.00 0.00 0.10 0.10 0.50 0.30 0.75

LQ-STAR OPTION LOST= '1'8

CLASS FI 0

EVENT NO.- 1= 4 2 4 3
EVENT NO.- 2= 2 1 2 2

CLASS FI 1

EVENT NO.- 1= 1 1 1 1
EVENT NO.- 2= 2 2 2 2
EVENT NO.- 3= 2 1 2 1
EVENT NO.- 4= 1 0 1 0

CLASS FI 2

EVENT NO.- 1= 0 0 0 0

Z(1)= 1.000000E+00 Z(2)= 5.000000E+00 Z(3)= 2.099995E+00 Z(4)= 1.000000E+00
W(1,1)= 1.000000E+00 W(1,2)= 1.000000E+00 W(1,3)= 1.000000E+00 W(1,4)= 1.000000E+00 W(2,1)= 1.000000E+00
W(2,2)= 1.000000E+00 W(2,3)= 2.529995E+00 W(2,4)= 1.000000E+00 W(3,1)= 1.000000E+00 W(3,2)= 1.000000E+00
W(3,3)= 1.000000E+00 W(3,4)= 1.000000E+00 W(4,1)= 1.000000E+00 W(4,2)= 1.000000E+00 W(4,3)= 1.000000E+00 W(4,4)= 1.000000E+00
THE FOLLOWING 2 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS 1

COMPLEX: \[(x_1 = 1, 2, (x_4 = 0, 1)]

COMPLEX: \[(x_2 = 2, (x_4 = 2)]

*** DELTA FOR THIS SET IS 0 ***

THE LARGEST STAR HAD 36 ELEMENTS; THE LARGEST INTERMEDIATE STAR HAD 36 ELEMENTS

THE FOLLOWING 1 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS 2

COMPLEX: \[(x_1 = 0)]

*** DELTA FOR THIS SET IS 0 ***

THE LARGEST STAR HAD 14 ELEMENTS; THE LARGEST INTERMEDIATE STAR HAD 27 ELEMENTS

THE FOLLOWING 2 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS 0

COMPLEX: \[(x_4 = 3)]

COMPLEX: \[(x_2 = 1, (x_4 = 2)]

*** DELTA FOR THIS SET IS 0 ***

THE LARGEST STAR HAD 20 ELEMENTS; THE LARGEST INTERMEDIATE STAR HAD 27 ELEMENTS

*** NORMAL TERMINATION ***
5. PROGRAM IMPLEMENTATION

5.1 General remarks

The basic algorithm used in the program for the synthesis of

\( \mathcal{N}_{\mathcal{H}} \) formulas is algorithm \( \mathcal{A}^2 \) described in papers \(^2,^3,^8\). In this

section, we will describe all the basic concepts, data structures and

the function of each procedure of the program. This description will be

sufficient for general understanding of the program and also can serve as

a guide for the detailed study of the program implementation. The complete

listing of the program is given in Appendix B of this paper.

5.2 Program representation of complexes and events

All events and complexes are represented in the program by

\( \text{PL/1} \) bit strings. Each variable \( x_i \) in an event or complex is assigned

a different substring of this bit string. Each position of the substring

corresponds to a value of this variable. Consequently, the length of the

substring assigned to variable \( x_i \) equals the number of values \( \text{NLEV}(i) \)

in the domain of this variable and the length of the entire string is

the sum of the lengths of each substring, i.e.

\[
\text{NV} = \sum_{i=1}^{\text{NLEV}(i)}
\]

\( (\text{NV} -- \text{number of variables}) \)

The leftmost position of the substring corresponds to value 0 and the

rightmost to value \( d_i \), i.e., maximum value of the given variable \( x_i \). If

in an event (complex) a variable \( x_i \) has (is referred to) value \( j \), then

\((j+1)\text{th}\) position of the substring is set to value 1. Remaining positions

are set to value 0. For example, the event

\( e = (0, 3, 2, 4) \)
from event space \( E(3, 4, 4, 5) \) (i.e., \( NLEV(1) = 3, NLEV(2) = 4, NLEV(3) = 4, NLEV(4) = 5 \)) is represented as:

\[
\begin{array}{cccc}
1 & 0 & 0 & 1 \\
x_1 & x_2 & x_3 & x_4
\end{array}
\]

Clearly, each substring in a string representing an event has exactly one bit set to 1, and each substring in a string representing a non-empty complex has at least one bit set to 1. If in a complex a given variable is absent, then the substring corresponding to this variable has all bits set to 1. For example, the complex

\[(x_1=2)(x_4=1,3)\]

is represented as:

\[
\begin{array}{cccc}
0 & 0 & 1 & 0 \\
x_1 & x_2 & x_3 & x_4
\end{array}
\]

5.3 Internal structure for storing intermediate stars

A star \( G(e|\Xi) \) of an event \( e \) against event set \( \Xi \) is generated in steps, each step corresponds to an event in \( \Xi \) (more precisely, each step involves the multiplication of the current set of complexes, called an 'intermediate star', by complexes resulting from the extension of \( e \) against the consecutive event \( e_i \) from \( \Xi \)). Thus, an intermediate star \( IG(e|\Xi) \) is a set of maximal complexes which cover \( e \) and do not cover some of the events from \( \Xi \) (rather than all events, as would be the case with \( G(e|\Xi) \)).

An intermediate star is represented by a doubly linked list, each element of which contains a pointer to the previous and to the next element in the list. In addition to a forward pointer (FPTR) and
a backward pointer (BPTR) each element of the list contains the bit
string representation of the complex (NUM), and a field (VAL) reserved
for the value of any criterion used to evaluate this complex during the
selection of quasi-extremal (i.e., selection of 'best complex').

The list can be accessed by a pointer which points to the first
element of the list (the head of the list) and a pointer which points to
the last element of the list (the tail of the list). In the process of
star generation three (doubly linked) lists are used:

1. a list representing the last intermediate star (with
   head FGPTR and tail BGPTR),
2. a list representing the intermediate star under
   construction (with head and tail pointers NEXTFG and NLGPT),
3. a list of free elements (with head and tail pointers
   NEXTFG and LASTFG), reserved for additional complexes.

All 3 lists are stored in an array structure G. A graphical
representation of storing an intermediate star within the structure G
follows:

![Graphical representation of storing an intermediate star within the structure G]
5.4 Structure of the program

The first set of PL/1 statements (approximately 150 statements) allocate the proper storage for all global variables. The input parameters are read and their values echoed on the printout. The type of each variable is recorded in a bitstring (TYPECV) where 'O'B in the i-th position indicates that variable \( x_i \) is of type INTERVAL, and '1' that it is of type FACTOR. The events are converted to bitstring representations as described above and loaded into an array E so that each row of E corresponds to an event set, and each column corresponds to an event in each set.

The next set of statements (approximately 40 statements) controls the selection of event sets for covering. The selection of these sets depends on the mode of operation of the program (specified by the parameter MODE) and on the order in which the sets are to be covered (as specified by the parameter QLIST). A row of the array E corresponding to the set of events to be covered is selected and the array FDOWN is loaded with the set(s) of events (and complexes, in the case of DC mode) against which the covers are to be generated. In addition, two bit arrays FNUMQ and FNUMSTAR are maintained. These arrays are in a one-to-one correspondence with the events which are to be covered (the columns in the selected row of E), (the array FNUMQ indicates elements which belong to the set \( E_1 \)) and the array FNUMSTAR indicates the elements which belong to the set \( E_p \) in the algorithm A presented in Figure 1 of paper2). A one ('1'B) in a position of FNUMQ indicates that the corresponding event of E belongs to \( E_1 \), i.e., it has not yet been covered by any quasi-optimal complex (Lq); a one ('1'B) in a position of FNUMSTAR indicates that the corresponding event belongs to \( E_p \), i.e., it has not yet been covered by any star (i.e., by any of the complexes of previously generated stars).
Next in the program are the loops controlling the selection of the events to be covered according to the arrays FLNOMQ and FLNOSTAR (approximately 50 statements). In the first loop, an event which has not been covered by any previous star (i.e., events of \( E_p \)) is selected according to FLNOSTAR. A star is generated for this event and the 'best complex' (quasi-extremal \( L^2 \)) is extracted by calls to the procedures STAR and BESTLQ. The resulting quasi-extremals are stored in a list MQ.INT along with a bit MQ.TRIM which indicates whether this complex was a result of a trimmed star or not. Once the set of events represented by FLNOSTAR (i.e., events of \( E_p \)) has been exhausted, the process is repeated with the set represented by FLNOMQ (i.e., set \( E_l \)). The number of passes through this second loop is calculated and stored in the parameter DELTA as an estimate of the distance of this set of formulas from the optimal set (see equation (12) in paper \(^2\)). The resulting set of complexes is printed by a call to the procedure FCOVER. Next a new set of events to be covered (row of \( E \)) is selected and the array FDOWN is loaded with new sets of events (and complexes in DC mode) against which a new cover is to be generated. After all event sets which were to be covered have been covered and the complexes printed out, the next set of parameters are read in. If no more parameters exist, the execution is stopped.

5.5 Basic procedures in the program

The remainder of the program consists of a set of procedures each of which performs a specific task as is explained below:

STAR -- This is the procedure which computes a star \( G(e|FDOWN) \) of an event \( e \) selected from the given row of the array \( E \) (representing event set to be covered) against the event set (and set of complexes in the DC mode) represented by \( FDOWN \). The star is generated in steps, each step corresponding to one element of \( FDOWN \) and producing an intermediate star. In each step, first the procedure
MIX is used which generates an 'elementary star' $G(e|L)$, where $L$ is the selected element of $FDOWN$, i.e., set of maximal complexes which cover $e$ and do not cover $L$. Now, the last intermediate star is multiplied by the 'elementary star' $G(e|L)$ and absorption rules are used before yielding the new intermediate star. The process ends when the last element of $FDOWN$ has been used. If any of the intermediate stars have more elements than $MAXSTAR$, a procedure TRIMSTAR is called.

MIX -- This procedure produces the star $T(e|L)$ where $L$ is an element (event or complex) in $FDOWN$. The stars produced by MIX are represented by the arrays INTP and POSP. Each complex of a star of an event against another event or complex involves only one variable (i.e., has only one literal), therefore in this type of star, only the values of one variable must be given in the representation of a complex in the star. In the program, INTP($i$) is a bitstring which represents the values of the variable whose index is stored in POSP($i$) (only non-empty complexes are retained) and NUM indicates the number of non-empty complexes which were placed into the star. The format of each of these complexes produced by MIX depends on the type of the variable which generates the complex. The procedure is as follows: the complement of the literal of the event or complex of $FDOWN$ against which a cover is being generated is computed. If this does not contain the value which the variable assumes in the chosen event of $E$ then the complex is thrown out (i.e., because it is empty). If the type of this variable is FACTOR, the complemented literal is used as the literal of the complex and loaded into the next place of INTP. Otherwise, the interval of this complement which contains the value of the corresponding literal of the chosen event $e$ is placed into INTP. In each case where there is a non-empty complex, the index of the variable which corresponds to the literal in INTP, is placed into POSP.

TRIMSTAR -- This procedure is invoked by the procedure STAR when the size of an intermediate star becomes greater than $MAXSTAR$. It reduces the star to the CUTSTAR best complexes according to criteria 1 and 2 defined above under the parameter CLIST. (These criteria measure the number of events in the set to be covered (indicated by FINOMC) and number of literals in a complex, respectively).

NUMLIT -- This procedure counts the number of literals in a given complex.
LCOVER -- This procedure counts the number of events covered by a given complex and not covered by any previously chosen complex (L^2) (i.e., the number of events indicated by FINOMG which are covered by the complex).

BESTLQ -- This procedure uses the criteria specified in the input to select the 'best' complex (quasi-extremal L^2) from the given star. The selection process is described in section 2 (parameter CLIST).

CRITVAL -- This is a function which returns the value of a criterion applied to a given complex. The value which is returned by CRITVAL for each criterion is described in section 2 (parameter CLIST).

RANGE -- is a procedure internal to CRITVAL used to determine the number of values of a variable in the given complex. 0.0 is returned if the variable does not appear in the complex (i.e., the variable accepts all possible values).

VARIANCE -- is a procedure internal to CRITVAL used to determine the mean deviation (described in criterion 7 in section 2 under the parameter CLIST) of the values of a variable in the given complex. 0.0 is returned if the variable does not appear in the complex.

PRTSTAR -- prints out the elements of a star.

PRTQLQ -- prints out trace information on L^2.

READVEC -- reads input events in vector format and converts them to the internal bit string representation.

READGAM -- reads input events in gamma format (i.e., as event numbers) and converts them to internal bit string representations.

CHECKE -- is an auxiliary procedure used by STAR in the process of combining stars (application of absorption laws).

CKBE -- is an auxiliary procedure used by STAR in the process of combining stars.
REMOVENV(REMOVENV) -- is used to remove a complex from the star during the generation and selection processes.

GETG -- obtains an element of G from the free list for use in a star.

RETURNG -- returns a list of elements of G to the free list.

EINCPLX -- is a function which determines if a given event in the set to be covered is covered by a given complex.

PCOVER -- prints out a cover in a readable format.

PFOREC -- is used by the initialization procedure to determine the amount of available core remaining. Note that it is dependent on the method of implementing pointer variables and on the layout of OS/360 control blocks.

5.6 Remarks about other related programs

The program AQVAL (AQ7) is currently the main and most developed program related to the computer implementation of VL₁. Other programs which are currently available include:

- CMHIBUS -- A program which can:
  A. Evaluate a given VL₁ formula (in which complexes are represented as bit strings) for any given event.
  B. Quantize a given set of variables according to a set of given thresholds.
  C. Select from the original set of events, subsets of events such that
     (i) they can involve only certain variables (out of all available variables)
     (ii) they can constitute any contiguous subset of the original set of events.
     (iii) gather statistics of performance of VL₁ formulas used as classification rules*.

*This last function is implemented in an extended form in the program CONTUS (see next).
• UNICLASS (AQ8) -- A special program developed for the synthesis of $V_{Lq}$ formulas from event sets which is able to create exact or approximate (with a user specified degree of approximation) covers of a set of events against its complement.\textsuperscript{21}

• AQVAL1 (AQ9) -- A program which synthesizes quasi-optimal $DVL_{l}$ formulas from non-optimal $DVL_{l}$ formulas.\textsuperscript{20}

• CONFUS -- A special program implementing various methods of evaluation of families of $DVL_{l}$ formulas and printing a confusion matrix which relates evaluated values to the correct known values for the formulas (appl: for testing formulas used as decision rules for pattern recognition problems).

• HISTOGRAM -- A program which prints a graph which illustrates the distribution of frequencies of individual values of a variable in an event set. (appl: determination of thresholds for quantization units in designing learning and testing for pattern recognition problems)

• SIML -- A program which detects symmetry in (incompletely specified) $V_{Lq}$ functions. (This program is still under development.)
REFERENCES


APPENDIX

Steps in input preparation

(Short form)

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>14</td>
</tr>
</tbody>
</table>

I. Specify control parameters:
- MODE ('IO')
- UNEQ ('VECTOR')
- TITL (3)
- NAGDR (150)
- CUTDR (50)
- NNE (500)
- EQ (25)
- ULRP ('18')
- SAVE ('O.E')
- QSTAT ('O.E')
- LTFACE ('O.E')
- SRFACE ('O.E')
- GST ('O.E')

II. Type semi-colon (;)

III. Type the title of the problem

IV. Specify data parameters:
- NSPEC TYPE TYPETLINSPEC
  (type of variables, e.g.,
  2 'VECTORS' 1, 4)
- NW IWEK (NW)
  (# of variables and size of their domains, e.g.,
  4 5, 3, 2, 4)
- NKL NEK (NKL)
  (# of event sets and their size, e.g.,
  3 5, 3, 2)
- CLIST
  (desired order of event sets, e.g.,
  1 0 2)
- NRMX CLIST (NRMX) TLIST (NRMX)
  (criteria list and tolerance list, e.g.,
  5 1, 2, 4 0, 6, 0.1)
- F
  (list of event sets, e.g.,
  0 0 1 0 2 1 3 2
  1 0 3 1 3 2 6 2)

Numbers in parenthesis are default values for parameters.

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
</tr>
</tbody>
</table>

Z
(list of costs for variables, e.g.,
Z(5) = 51)

W
(list of weights for events, e.g.,
W(1,3) = 5  W(0,1) = 2)

List of User Optimization Criteria

<table>
<thead>
<tr>
<th>Criterion #</th>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Number of terms in a formula (t)</td>
</tr>
<tr>
<td>2</td>
<td>Number of selectors in a formula (s)</td>
</tr>
<tr>
<td>3</td>
<td>Cost of variables in a formula (z)</td>
</tr>
<tr>
<td>4</td>
<td>Degree of generalization (g)</td>
</tr>
<tr>
<td>5</td>
<td>Weight of events covered by a formula (w)</td>
</tr>
<tr>
<td>6</td>
<td>Length of references (l)</td>
</tr>
<tr>
<td>7</td>
<td>Relative scope of references (r)</td>
</tr>
</tbody>
</table>
APPENDIX B

PROGRAM LISTING
SOURCE LISTING

STMT LEV NT

AQ7: PROC OPTIONS (MAIN); /* AQ-VERSION AQ-7 */

/* THIS PROCEDURE DETERMINES A SET OF COMPLEXES WHICH WILL
COVER FILE BUT WILL SPECIFICALLY EXCLUDE FOE.

A STAR IS STORED IN THE STRUCTURE G. EACH ELEMENT OF G
CONTAINS THE FOLLOWING ITEMS:
THE DEFINITION OF THE COMPLEX,
A FORWARD POINTER, AND
A BACKWARD POINTER.

ELEMENTS OF G ARE LINKED TOGETHER VIA THEIR FORWARD AND
BACKWARD POINTERS INTO THREE DISTINCT LISTS:
A FREE LIST WHICH CONTAINS ALL UNUSED ELEMENTS;
A CURRENTLY COMPLETE INTERMEDIATE STAR, AND
THE NEXT (USUALLY INCOMPLETE) INTERMEDIATE STAR.

THREE PAIRS OF EXTERNAL POINTERS ARE USED TO DELINEATE
THE THREE LISTS. THE FIRST PAIR, NEXTFG AND LASTFG, POINT
TO THE BEGINNING AND ENDING ELEMENTS OF THE FREE LIST.
A SECOND PAIR, FPTR AND LPTR, POINT TO THE BEGINNING AND
ENDING ELEMENTS OF THE CURRENTLY COMPLETE INTERMEDIATE STAR.
THE THIRD PAIR, NFPTOR AND LFPTOR, POINT TO THE BEGINNING
AND ENDING ELEMENTS OF THE NEXT INTERMEDIATE STAR WHICH IS
BEING FORMED.

IN GENERAL THE BACKWARD POINTER OF THE FIRST ELEMENT IN
EACH LIST WILL POINT TO NULL (WHICH IS DENOTED AS A ZERO)
AS WILL THE FORWARD-POINTER OF THE LAST ELEMENT OF EACH LIST.
THE EXCEPTION TO THIS IS THE LINKS OF THE LAST ELEMENT OF
THE NEXT INTERMEDIATE STAR AND THE FIRST ELEMENT OF THE FREE
LIST. THESE ELEMENTS ARE LINKED TOGETHER UNTIL THE NEXT
INTERMEDIATE STAR IS COMPLETED IN ORDER TO EXPEDITE THE
PROCESS OF THE GROWING OF A STAR.

EVENTS ARE STORED IN A BI-UNARY BIT STRING FORMAT.
FOR EXAMPLE, IF NV = 4, NLEV = 6, AND AN EVENT E = (5,0,3,1),
THEN THE INTERNAL BIT STRING WOULD BE STORED AS:
1000000000000100000000000010B.

INTERVALS ARE PAIRS OF VECTORS AND ARE STORED TO SHOW THE
UPPER AND LOWER BOUND OF EACH VARIABLE. THIS AN INTERVAL
WHOSE UPPER BOUND VECTOR WAS = (4,5,3,2) AND
WHOSE LOWER BOUND VECTOR WAS = (2,0,3,1) WOULD BE STORED AS
0110011111100000010B.

ALL INPUT IS FREE FORMAT IN THE FOLLOWING ORDER:
INGE, NMQ, MAXSTAR, CUTSTAR, LQFAC, STRACE, QST, QLOT, INFORM,
TITLE) ALL OPTIONAL & IN PLI DATA FORMAT, A SEMICOLON (MUST APPEAR
DUE TO THE PREVIOUS VARIABLES BEING READ IN IN DATA FORM).
TITLE LINES FOR THIS
NSPEC, TYPE, NUMBERS OF VARIABLES TO BE SPECIFIED AS IN
TYPE. ALL OTHER VARIABLES ARE COVERED BY THE
OTHER TYPE OF COVER);
NV, NL, NI, NO, FLE, FDE,
NCE, NMO, MAXSTAR, CUTSTAR, & TITLE ARE INTEGER VALUES IN DATA
FORM,
NV, NL, NI, NO ARE INTEGER VALUES IN LIST FORMAT,
LQTRACE, STRACE, QST, OLQ, E SAVE ARE BIT STRINGS OF LENGTH 1
IN DATA FORMAT,
INFORM IS A CHARACTER STRING IN DATA FORMAT.
TITLE LINES ARE ANY CHARACTERS STARTING IN COLUMN 1 OF THE LINE
FOLLOWING THE SEMICOLON AND CONTINUING FOR TITLE NUMBER
OF LINES.
FLE AND FDE ARE ENTERED IN ONE OF TWO FORMS: GAMMA OR VECTOR.
REFER TO THOSE DEFINITIONS FOR DETAILS.

TWO EXAMPLES OF INPUT DATA FOLLOW ON THE NEXT SIX LINES:
NGE = 100, OLQ = '1'B, INFORM = 'VECTOR';
2 'FACTORS' 2,3,
3, 4, 2, 3,
0, 1, 2, 3, 2, 3, 10 3 2 0 0 3, 3
NHQ = 20, STRACE = '0'B INFORM = 'GAMMA';
0 'INTERVALS' 3, 4, 2, 3,
6, 59 19, 53, 15
*/

DECLARE
(LQTRACE, STRACE, QST, OLQ, ESAVE, BIT(1)) ALIGNED STATIC,
SAVELO BIT(1) ALIGNED STATIC,
MODE CHAR(2) STATIC,
(INFORM CHAR(6), LINE CHAR(132) VARYING) STATIC,
(NV, NL, LEN, NCE, NMO) FIXED BINARY(15) STATIC,
(I, J, K, L, M, N) FIXED BINARY(15) STATIC,
(LQTRACE, OLQ, STRACE, NHQ) FIXED BINARY(15) STATIC,
(NEXTFG, LASTFG) FIXED BINARY(15) STATIC,
(NMPT, DELTA, HGPTS) FIXED BINARY(15) STATIC,
(TIL, LG, NFL, NFILQ) FIXED BINARY(15) STATIC,
(NFST, LINES, LINEAL, LS, LVN) FIXED BINARY(15) STATIC,
(MAXSTAR, CUTSTAR, NUMTHRM, NUMTHRM) FIXED BINARY(15) STATIC,
TITLE FIXED BINARY(15) STATIC,
CHARBUF CHARACTER(80) STATIC,
COVER FILE STREAM OUTPUT EXTERNAL &
DCL(TYPEDV64) BIT(1), LST BIT(1),
NSPEC FIXED BINARY(15),
TYPE CHAR(9) STATIC,
DEFINITION OF VARIABLES DECLARED IN THIS BLOCK:

NVP    - (BIT POSITION) STARTING BIT POSITION OF THE DESIRED VARIABLE WITHIN THE TOTAL BIT STRING
CHARBUF - CHARACTER BUFFER USED AS TEMPORARY STORAGE FOR THE TITLE OF THE PROBLEM DATA
COVER  - OUTPUT FILE INTO WHICH COVERING COMPLEXES WILL BE WRITTEN IF SAVE = '1'
CUTSTAR - NUMBER OF ELEMENTS TO WHICH THE STAR SHOULD BE REDUCED (CUT) IF IT EXCEEDS MAXSTAR ELEMENTS
DELTA  - AN ESTIMATE OF THE DIFFERENCE BETWEEN THE NUMBER OF COMPLEXES IN THIS COVER, AND IN A MINIMAL ONE
E      - TWO-DIMENSIONAL BIT(LEN)-ARRAY CONTAINING ALL ELEMENTS OF THE CLASSES F-1,...,F-MAXCL
FGPTR  - (FIRST G POINTER) POINTS TO THE FIRST COMPLEX OF THE CURRENT "STAR"
FMIX   - (FIRST MIX CALL) USED TO DIFFERENTIATE BETWEEN FIRST (FMIX = '1') OR LATER (FMIX = '0') CALL TO THE PROCEDURE MIX IN EACH STAR. IT IS SET ONLY IF THERE IS AN ERROR IN THE INPUT DATA.
GAMMA  - INPUT FORMAT FOR TRUE AND FALSE VECTORS, WHERE EACH VECTOR IS REPRESENTED AS A UNIQUE NUMBER J DETERMINED AS FOLLOWS:
J = KINV ;
DO I = 1 TO NV-1 ;
   PROD = 1 ;
   DO K = 0 TO K-1 ;
      PROD = PROD * NLEV(NV-K) ;
   END ;
   J = J + XINV(I) * PROD ;
END ;
OR WHEN NLEV(1)=NLEV(2)=...=NLEV(NV-1), THEN
J = 0 ;
DO I = 0 TO NV-1 ;
   J = J + XINV(I) * NL*II ;
END ;
GPTR   - (G POINTER) POINTS TO THE COMPLEX CURRENTLY BEING USED
I      - USED AS A GENERAL PURPOSE INDEX PLUS AS AN INDEX ON THE COMPLEXES OF THE STAR
IG     - USED AS AN INDEX FOR CONSTRUCTING THE INITIAL STAR
IKL    - USED AS AN INDEX OVER THE KL ARRAY
INFORM - (INPUT FORMAT) FORMAT FOR THE TRUE AND FALSE EVENTS CAN EITHER BE 'GAMMA' OR 'VECTOR'
IO     - USED AS AN INDEX OVER THE FIE ARRAY
II     - USED AN AN INDEX OVER THE FIE ARRAY
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IIL</td>
<td>INDEX USED TO POINT TO THE ELEMENT OF F1 LAMBDA FOR WHICH A STAR IS BEING GENERATED</td>
</tr>
<tr>
<td>J</td>
<td>USED AS AN INDEX OVER THE NEW COMPLEXES THAT ARE TO BE MULTIPLIED TIMES THE CURRENT STAR</td>
</tr>
<tr>
<td>K</td>
<td>CURRENT 0-LIST ELEMENT: K=OLIST(OLP(TR))</td>
</tr>
<tr>
<td>LASTFG</td>
<td>(LAST FREE G) POINTS TO THE LAST G(E) ELEMENT IN THE &quot;FREE&quot; LIST</td>
</tr>
<tr>
<td>LEN</td>
<td>(LENGTH) LENGTH OF THE UNARY BIT STRING USED TO REPRESENT ELEMENTS</td>
</tr>
<tr>
<td>LGPTR</td>
<td>(LAST G POINTER) POINTS TO THE LAST COMPLEX OF THE CURRENT &quot;STAR&quot;</td>
</tr>
<tr>
<td>LNEAL</td>
<td>LARGEST NUMBER OF ELEMENTS IN ANY INTERMEDIATE STAR</td>
</tr>
<tr>
<td>LNES</td>
<td>LARGEST NUMBER OF ELEMENTS IN EACH INTERMEDIATE STAR</td>
</tr>
<tr>
<td>LQ</td>
<td>POINTER TO AN ENTRY OF G(E) CONSIDERED &quot;BEST&quot; OF THE STAR ACCORDING TO GIVEN CRITERIA</td>
</tr>
<tr>
<td>LSS</td>
<td>LARGEST STAR SIZE IN NUMBER OF ELEMENTS</td>
</tr>
<tr>
<td>MAXSTAR</td>
<td>LARGEST NUMBER OF ELEMENTS THAT A STAR IS ALLOWED TO HAVE BEFORE TRIMMING IS APPLIED</td>
</tr>
<tr>
<td>MAXCL</td>
<td>HIGHEST CLASS NUMBER</td>
</tr>
<tr>
<td>MODE</td>
<td>VL OR IC MODE</td>
</tr>
<tr>
<td>NECMAX</td>
<td>NCL*NMAX, I.E. HIGHEST DIMENSION OF ARRAY FDOWN</td>
</tr>
<tr>
<td>NCL</td>
<td># OF CLASSES (=MAXCL+1)</td>
</tr>
<tr>
<td>MOPTR</td>
<td>(MC POINTER) POINTS TO THE LAST COMPLEX IN MQ</td>
</tr>
<tr>
<td>NES</td>
<td>NUMBER OF ELEMENTS IN A STAR</td>
</tr>
<tr>
<td>NEXTG</td>
<td>(NEXT FREE G) POINTS TO THE NEXT G(E) ELEMENT IN THE &quot;FREE&quot; OR UNUSED LIST OF G(E)</td>
</tr>
<tr>
<td>NE</td>
<td>(NUMBER OF ELEMENTS IN F) NE(I)=# OF ELEMENTS IN CLASS F=I, FOR I=0,...,MAXCL</td>
</tr>
<tr>
<td>NMAX</td>
<td>LARGEST COMPONENT OF NE</td>
</tr>
<tr>
<td>NFOPTR</td>
<td>(NEXT FIRST G POINTER) POINTS TO THE FIRST COMPLEX OF THE NEW STAR</td>
</tr>
<tr>
<td>CVNOSTAR</td>
<td>NUMBER OF EVENTS NEVER COVERED BY ANY STAR (I.E. NUMBER OF EVENTS IN F1NOSTAR)</td>
</tr>
<tr>
<td>NFIL0</td>
<td>NUMBER OF F1 ELEMENTS COVERED BY PRESENT LQ</td>
</tr>
<tr>
<td>NGE</td>
<td>(NUMBER OF G(E) ELEMENTS) TOTAL NUMBER OF G(E) IN THE FREE LIST PLUS THE USED LIST</td>
</tr>
<tr>
<td>NLEV</td>
<td>NUMBER OF LEVELS FOR EACH VARIABLE</td>
</tr>
<tr>
<td>NLGPTR</td>
<td>(NEXT LAST G POINTER) POINTS TO THE LAST COMPLEX OF THE NEW STAR</td>
</tr>
<tr>
<td>NHQ</td>
<td>(NUMBER OF MQ) MAXIMUM NUMBER OF COMPLEXES WHICH CAN BE STORED IN LIST MQ</td>
</tr>
<tr>
<td>REMUNCV</td>
<td>NEW NUMBER OF ELEMENTS IN F1NOSTAR THAT ARE UNCOVERED</td>
</tr>
<tr>
<td>NFOIL0</td>
<td>NUMBER OF ELEMENTS IN THE ORIGINAL F1 COVERED</td>
</tr>
</tbody>
</table>
NUM - number of complexes resulting from a call to MIX
NSPEC - no. of variables specified to be covered by
factors resp. intervals (input data)
NUMSTRA - number of stars which have been trimmed
NUMTRIM - number of times a given star has been trimmed
NV - number of variables
ONES - bit string of all 1 bits
QLQT - (quick lo trace) variable which allows a minimal
trace of the lo determination process. size of the
star, number of element of fl covered by LO, and
count of remaining elements of fl通知 star are printed after
each lo is determined when QLQT=1'B.
This trace is suppressed if QLQT=0'B.
QST - (quick star trace) variable which allows a
minimal trace of the star generation process
(QST=1'B). a message will be printed at the
completion of the multiplication resulting from
each element of foe.
This trace is suppressed if QST=0'B.
SAVE - (save cover data) set to 1'B to cause complexes
in the solution cover to be put into file cover.
If SAVE=0'B nothing is placed into file cover.
STRACE - (star trace) variable which allows a printed
trace of the star generation process (STRACE=1'B).
The results of the calls to MIX, all new generated
complexes, the value of the complexes of the star
at the end of each step of star generation, and
the calls to getg and returning are printed.
No trace is printed if STRACE=0'B.
TITLE - if TITLE > 0 then it indicates the number of input
lines in the title. if TITLE <= 0 then there is
no title in the input data stream.
TYPE - (type of cover) either 'intervals' or 'factors'
TYPECVR - (type of cover) array of 64 bits, where
0'B = interval covering, 1'B = factor covering
attempted for this variable
VECTOR - input format for true and false vectors, where
each component of each vector is in the input i.e.,
FILE(S) = (3,2,1,4) would be listed on the input line as
3 2 1 4 2 3
ZEROS - bit string of all 0 bits
(4 A, SKIP, COL(2), 2 A):
END;
ON ENDFILE(SYSIN) GO TO FINISH;
NEWDATA:
CUTSTAR=50; MAXSTAR=150;
NGE = 200 ;
NMQ = 25 ;
MODE=1C*;
LOGLEV=STRACE, QLOT, SAVETHQ, SAVELQ = '0'B;
TITLE = 3 ; LST='1B';
GET DATA(NGE, NMQ, LOGLEV, SAVETHQ, SAVELQ, MAXSTAR, CUTSTAR, INFORM, TITLE, LST);
PUT PAGE;
IF TITLE>0 THEN DO:
    PUT SKIP(2);
    GET SKIP(1);
    DO 1=1 TO TITLE;
1    GET EDIT(CHARBUF)(A(80));
    PUT SKIP(1) EDIT (CHARBUF)(COLUMN(20), A);
END;
END;
PUT SKIP(3) EDIT
('SPACE ALLOCATED FOR G(E) IS', NGE) (A,F(5));
('SPACE ALLOCATED FOR MQ IS', NMQ) (COLUMN(50), A,F(5));
PUT SKIP(2) EDIT
('MAXIMUM STAR SIZE BEFORE TRIMMING WILL BE DONE =',
MAXSTAR,' THEN IT WILL BE CUT TO', CUTSTAR,' THEN IT WILL BE CUT TO',
A,F(6),A,F(6),A,A);
IF INDEX('IC',MODE)=0 THEN
    DOX PUT SKIP LIST(' ICG MODE ASSUMED'); MODE='IC';END;
PUT SKIP(2) EDIT
('LOG TRACE =', LOGLEV, 'STARC TRACE =', STRACE,
'QUICK LOG TRACE =', QLOT, 'QUICK STAR TRACE =', QST,
'SAVE COVER DATA =', SAVETHQ, 'SAVE QLOT =', SAVELQ) (61,A,F(2),X(13));
PUT SKIP(2) EDIT('INPUT FORMAT IS', INFORM)(A,aA);
/* INTERVALS OR FACTORS */
GET LIST(NSPEC, TYPE);
TFPCOV='0'B;
IF NSPEC > 0 THEN DO I=1 TO NSPEC;
    GET LIST(I);
    TFPCOV(I)='1B';
END;
IF SUBSTR(TYPE,1,3)='INT' THEN TFPCOV = ~TFPCOV;
ELSE IF SUBSTR(TYPE,1,3) = 'FAC' THEN PUT SKIP(2) LIST
(*** INVALID COVER TYPE SPECIFIED - FACTOR ASSUMED ***):

GET LIST(NV1);

IF NSPEC=0 1 NSPEC=NV THEN DO:
  LINE='ALL VARIABLES WILL BE COVERED BY *';
  IF TYPcov(I) THEN LINE='LINE'|'FACTORS';
  ELSE LINE=LINE'|'INTERVALS';
  PUT SKIP(2) LIST(LINE);
END;

ELSE DO:
  PUT SKIP(2) LIST('THE FOLLOWING VARIABLES ARE ' |
  ('COVERED BY FACTORS');
  DO I=1 TO NV; IF (50*(I-1)+4)>120 THEN PUT SKIP DATA;
    IF TYPcov(I) THEN PUT EDIT (1)
    (COLUMN(50*(I-1)+4),F(3));
  END;

  PUT SKIP LIST('THE FOLLOWING VARIABLES ARE ' |
  ('COVERED BY INTERVALS');
  DO I=1 TO NV; IF (50*(I-1)+4)>120 THEN PUT SKIP(1) DATA;
    IF -TYPcov(I) THEN PUT EDIT(I)
    (COLUMN(50*(I-1)+4),F(3));
  END;

  PUT SKIP(2) EDIT('NUMBER OF VARIABLES = ',NV1(A,F(4)));

IF CUTSTAR > MAXSTAR THEN
  DO:
    PUT SKIP(3) LIST
    (** CUTSTAR CANNOT BE GREATER THAN MAXSTAR **):
  END;

  IF SAVE THEN OPEN FILE(COVER) LINESIZE(80) ;

MAINBLK: BEGIN REORDER:

DECLARE (NLEV(NV),NVP(NV)) FIXED BINARY(15);

GET LIST(NL);

PUT SKIP(2) EDIT('NUMBER OF LEVELS FOR EACH VARIABLE = ',NL)
(A,(NV1)(F(3)));

LEN,NLMAX=0;
DO I=1 TO NV;
  NVP(I)=LEN+1;
  LEN=LEN+NLEV(I);
IF NLMAX < NLEV(1) THEN NLMAX = NLEV(1);
END;

GET LIST(NCL,(NE(1) DO I=0 TO NCL-1));
MAXCL = NCL-1;

PUT SKIP(2) EDIT('NUMBER OF EVENTS',
'SPECIFIED FOR EACH CLASS', 'CLASS #EVENTS',
(I,NE(1) DO I=0 TO MAXCL)
(A,A,SKIP(1),COLUMN(8I),A,SKIP(1),
(I0I) (42F10),SKIP(1));

NEMAX, NLMAX = 0;
DO I=0 TO MAXCL;
IF NEMAX < NE(1) THEN NEMAX = NE(1);
NECLMAX = NECLMAX + NE(1);
END;

BEGIN REORDER:

DECLARE (EID: MAXCL, 1: NEMAX),
    FDOWN(1: NECLMAX) BIT(LEN) ALIGNED,
    TAINT BIT(LEN) ALIGNED,
    (OLIST(1: NCL), CLIST(1: NCL), NCRIT) FIXED BINARY(15),
    (OLPRT, OMPRT) FIXED BINARY(15),
    TLIST(17) DEC FIXED(5,2),
    RENUMCOV FIXED DECIMAL(3) STATIC,
    ZEROS BIT(LEN) INIT(0*8),
    ONESINIT(1) BIT(1) INIT(1*8),
    ONESBIT(LEN) INIT(LEN(1)*1B),
    ONESBIT(LEN) DEFINED ONESINIT,
    (INDEP(30), TOTCOV(30), NEMCOV(30), NUMCOV(30))
    FIXED BIN(15), (COMCOV, ISPL) FIXED BIN(15),
    1 CNGE),
    2 INT BIT(LEN) ALIGNED, /* INTERVAL */
    2 VAL DEC FLOAT, /* CRITERION VALUE */
    2 FPT Fixed BINARY(15), /* FORWARD POINTER */
    2 BPTR Fixed BINARY(15), /* BACKWARD POINTER */
    1 MQ(NWO),
    2 INT BIT(LEN) ALIGNED, /* INTERVAL */
    2 TRIM CHARACTER(1), /* TRIMMING INFO */
    (INTP(NV), TINT, TAND) BIT(NLMAX) ALIGNED,
    (MNV, FINDQAD(NEMAX), FINDQF(NEMAX), FINDSTAR(NEMAX)) BIT(1),
    (PDPENV, KLN(NV, NGE(2)), KEIN, NGE(2), KNL(NV, NGE(2)),
    KLC(NV), KEG(NV), KBC(NV)) FIXED BINARY(15),
    (WID(MAXCL), 1: NEMAX), Z(1: NV) DEC FLOAT;
/* DEFINITION OF VARIABLES DECLARED IN THIS BLOCK:
BPTR - (BACK POINTER) POINTS TO THE ELEMENT OF G PRECEDING IT IN THE CURRENT LIST

CLIST - (CRITERIA LIST) LIST OF NCRIT CRITERIA TO SELECT AN LG FROM THE CURRENT STAR IN PROCE BESTLG

FINOMQ - BIT-ARRAY CONTAINING ELEMENTS OF E(k,*j) WHICH HAVE NOT BEEN COVERED YET BY A QUASI-EXTREMAL.
I.E. FINOMQ(*j)=1/0 <-> E(k,*j) NOT YET COVERED.

FINOSTAR - BIT ARRAY DENOTING ELEMENTS OF E(k,*j) WHICH HAVE NEVER BEEN COVERED BY ANY STAR; I.E., FINOSTAR(*j)=1/0 MEANS THAT E(k,*j) HAS NEVER BEEN COVERED BY ANY STAR.

FPTR - (FORWARD POINTER) POINTS TO THE ELEMENT OF G FOLLOWING IT IN THE CURRENT LIST

FDOWN G - ARRAY OF ALL ELEMENTS AGAINST WHICH WE ARE COVERING

INT - LIST USED FOR STORING INTERVALS OF STARS

INTP - BIT-UNARY BIT STRING USED TO REPRESENT AN INTERVAL

KB - ARRAY WHICH STORES POINTERS TO CONDITION 4 INTERVALS IN THE NEW STAR; THESE INTERVALS CAN EITHER COVER NEXT CONDITION 4 INTERVALS OR BE COVERED BY NEXT CONDITION 4 INTERVALS.

KBC - KBC(*) IS THE COUNT OF THE NUMBER OF POINTERS IN KB(*)

KE - ARRAY WHICH STORES POINTERS TO EARLIER INTERVALS IN THE NEW STAR WHICH COULD POSSIBLY BE COVERED BY THE NEXT INTERVALS TO BE GENERATED IN THIS STEP.

KGC - KGC(*) IS THE COUNT OF THE NUMBER OF POINTERS IN KC(*)

KL - ARRAY WHICH STORES POINTERS TO INTERVALS IN THE NEW STAR WHICH COULD POSSIBLY COVER INTERVALS FOR THIS NEW STAR WHICH WILL BE GENERATED LATER IN THIS STEP.

KLC - KLC(*) IS THE COUNT OF THE NUMBER OF POINTERS IN KL(*)

N - AN ARRAY USED TO MARK THOSE LITERALS GENERATED FROM MIX WHICH CAUSE CONDITION 2 IN THE MULTIPLICATION PROCESS.

MQ - LIST USED FOR STORING INTERVALS REQUIRED FOR A COVER

NCRIT - (NUMBER OF CRITERIA) # OF CRITERIA SPECIFIED IN THE C-LIST.

NODOWN - # OF ELEMENTS IN FDOWN

DLIST - (ORDER LIST) LIST SPECIFYING THE ORDER IN WHICH THE CLASSES F-0, ..., F-MAXCL ARE CONSIDERED FOR COVERING.

OLPTR - (O-LIST POINTER)

OLPSTRP - (OLD O-P-POINTER)

P3SP - POSITION OF THE LITERAL; I.E., THE VARIABLE NUMBER (RANGES FROM 1 TO NV)

TAINT - TEMPORARY BIT STRING USED FOR STORING AN INTERVAL

TAND - TEMPORARY BIT STRING USED FOR STORING THE RESULTS OF ANDING TWO BIT STRINGS
TINT      =  TEMPORARY BIT STRING USED FOR STORING A LITERAL
TRIM      =  CHARACTER USED TO INDICATE THAT THIS INTERVAL
            RESULTED FROM A STAR WHICH WAS TRIMMED (TRIM="#")
            OR A STAR WHICH WAS NOT TRIMMED (TRIM=" ")

*** */

89 3 0    FDOWN = ZEROS:
90 3 0    PUT SKIP(2) EDIT
         (** AMOUNT OF UNUSED CORE = "FRECORE" **)
         (COLUMN(35),A,F(4),A):
91 3 0    PUT SKIP(2):
92 3 0    /* INITIALIZE VARIOUS VARIABLES AND CONSTANTS */

93 3 0    NEXTFG = 1:
94 3 0    LASTFG = NGE:
95 3 1    DO IG = 1 TO NGE:
96 3 1    G.PPTR(IG) = IG + 1:
97 3 1    G.OPTR(IG) = IG - 1:
98 3 1    END:
99 3 0    FPTR(NGE) = 0:
100 3 0   GET LIST((OLIST(I)) DO I=1 TO NCL));
101 3 0   GET LIST((CLIST(I)) DO I=1 TO NCRIT));
102 3 0   PUT SKIP(2) EDIT
         (**OLIST=' ',OLIST,'NCRIT=' ,NCRIT,'CLIST=' ,
         (CLIST(I)) DO I=1 TO NCRIT))
         (A ,INCL)(X(2),F(2),SKIP(2),A,F(5),SKIP(1),
         A ,NCRIT)(F(5),X(3))):
103 3 0   GET LIST((TLIST(I)) DO I=1 TO NCRIT));
104 3 0   PUT SKIP(2) EDIT('LIST=',
         (TLIST(I)) DO I=1 TO NCRIT)) (A ,INCRIT)(F(0,2)):

105 3 0    /* GET INPUT EVENTS */
106 3 0    IF INFORM = 'VECTOR' THEN CALL READVEC;
107 3 0    ELSE IF INFORM = 'GAMMA' THEN CALL READGAM;
108 3 0    ELSE DO:
109 3 1    PUT SKIP(3) LIST
         (** INVALID EVENT FORMAT SPECIFIED ***)
         GO TO FINISH:
110 3 1    END:
111 3 0    DLPTR = 1; MOPTR = 0;
112 3 0    NDOWN = 0:
113 3 0    IF MODE = 'VL' THEN
114 3 0    DO I = NCL TO 1 BY -1:
115 3 1    II = OLIST(I);
FINOSTAR,FINOMQ="IB; 
CVNOSTAR,NOMILQ,NUMSTRM,LNEAL,LSS="O;  
/* FIND NEXT ELEMENT OF FINOSTAR WHICH HAS NOT YET BEEN COVERED */

DO ILL=1 TO NE(K);  
  IF FINOSTAR(ILL) THEN DO;  
    CALL STAR;  
    CALL BESTLQ ;  
    IF QLOT THEN CALL PRTOLO;  
    IF QLOT LOR TRACE THEN DO;  
      /\ COUNT # UNCOVERED ELEMENTS  
      REMUNCOV=REMUNCOV+1;  
    END;  
    IF FINOSTAR(I) THEN  
      REMUNCOV=REMUNCOV+1;  
  END;  
END;  
END:  
END:  
/* FIND NEXT ELEMENT OF FINOMQ WHICH HAS NOT YET BEEN COVERED */

DO ILL=1 TO NE(K);  
  IF FINOMQ(ILL) THEN DO;  
    CALL STAR;  
    CALL BESTLQ ;  
    IF QLOT THEN CALL PRTOLO;  
    DELTA = DELTA + 1 ;  
  END;  
END:  
/* PRINT OUT CARTESIAN COMPLEXES COVERING CLASS K */
HOPTS=HOPTR=OMOPTR:
PUT SKIP(5) EDIT(*THE FOLLOWING*,HOPTS;  
* CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS*,K;  
* COV NEW IND TOT*)  
(A,F(3),A,F(3)),(N5),A;  
IF SAVE THEN PUT FILE(COVERI SKIP EDIT(NL,HOPTS;  
* DELTA=*,DELTA=(DELTA,3,F(4),F(4),A);  
ISPL=1:  
CALL TRIPLE;  
DO I=OMOPTR+1 TO HOPTR;
CALL PRCOVER(MQ,INT(I)):

ISPL=ISPL+1;
IF SAVE THEN
PUT FILE(COVER) SKIP LIST(MQ,INT(I));
PUT SKIP;
END;
PUT SKIP(3) EDIT('*** DELTA FOR THIS SET IS',DELTA,'***')(A,F(3),A);
PUT SKIP(3) EDIT('THE LARGEST STAR HAD',LSS,' ELEMENTS; The LARGEST INTERMEDIATE STAR HAD',LEAIL,' ELEMENTS')(A,F(5),A,F(5),A);
IF NUMSTRM > 0 THEN PUT SKIP(3) EDIT
   ('NOTE: INTERVALS RESULTING FROM TRIMMED STARS.'),
   ('TOTAL # STARS TRIMMED ='),(NUMSTRM)(A,A,F(4));
OLPTR=OLPTR+1;
END NEXTOLP;
NTMIC:PUT SKIP(3) LIST('*** NORMAL TERMINATION ***');
GO TO NEWDATA;
/

******************************************************************
******************************************************************
FUNCTION REORDER;

/* THIS PROCEDURE DETERMINES THE STAR FOR ELEMENT ILL
   OF ARRAY E(K,*). UPON RETURN THE STAR WILL BE STORED
   BETWEEN POINTERS FGPTR AND LGPTR IN THE G LIST. */

NUMSTRM = 0;
/* START FORMING G(E) BY (E MIX (COMEPL(E)))) */
CALL MIX(E(K,ILL),FDOHN(1),INTP,POSP,NUM);
IF STRACE THEN PUT SKIP(2) EDIT
   ('START G(E)'),(INTP(IG),POSP(IG))
   DO IG = 1 TO NUM }
   (A,(NUM)(X(4)+A,F(4))) ;
IF NUM = 0 THEN
   IFMIX = '1'B;
   GO TO TEOF ;
   END :
   DO IG = 1 TO NUM :
   CALL GETG(LGPTR) ;
   IF IG = 1 THEN FGPTR = LGPTR ;
   G.INT(LGPTR) = ONES ;
SUBSTR(G,INTP(LGPTR),NVP(POSPIG1)),NLEV(POSPIG1))
   = INTP1(I3);
   IF STRACE THEN DO:
      PUT SKIP EDIT
         ('FIRST INTERVAL:' (A));
      CALL PRCOVER(G,INTP(LGPTR));
   END;
   END;
   END 1:
   G,FPTR(LGPTR) = G,FPTR(FG PTR) = 0 ;
   NES, LNES = NUM 1;
   IF GST THEN PUT SKIP(2) EDIT
      ('*FINISHED MLT. FOR FOE(' 1'),
      '*STAR SIZE = ' NES,
      '*LARGEST INTERMEDIATE STAR SIZE = ' LNES)
      (A,X(8),XI(5)X(4),A,F(5));

   /* THIS COMPLETES E MIX (COMPL(E1))
   NOW FORM PRODUCT OF THIS WITH (E MIX (COMPL(E2))
   (E MIX (COMPL(E3))) ...
   */

KPROD:  DO IO = 2 TO NDOWN:
   NES = 0 1;
   CALL MIX(E(K1),I1L),FDOWN(IO),INTP,POSPI,NUM 1;
   IF STRACE THEN PUT SKIP(2) EDIT
      ('NEXT G(E) = ' INTP (I1L),POSPI)
      (IO = 1 TO NUM )
      (A,NUM,X(4),R,F(4));
   IF NUM = 0 THEN DO:
      FMIX = '0'B;
      GO TO TEOF ;
   END:
   KLG,KGC,KEC = 0 1;
   NFGPTR = 0 1; /* INITIALIZE NEXT FIRST G PTR */
   GPTR = FG PTR ; /* START LOOP OVER 'IN' */
   ZEROM:  DO J = 1 TO NUM ; /* ZERO H ARRAY */
      MI(J) = '0'B;
   END:

   /* NOW RUN THROUGH ALL INTERVALS RESULTING FROM
   LAST CALL TO MIX... */
   OVERJ:  DO J = 1 TO NUM :
      TINT = SUBSTRIG(INTP(GPTR),NVP(POSPI(J)),
      NLEV(POSPI(J1)));
      TAND = TINT & INTP(J);
      /* TEST FOR CONDITIONS 1 AND 3 */
      IF TAND = TINT THEN DO: /* CONDITION 1 OR 3, SO
STORE INTERVAL G|G|PTR) INTO NEW G LIST. */
CALL GETG(NLG|G|TR) ;
IF NFG|G|TR = 0 THEN
   DO :
      NFG|G|TR = NLG|G|TR ;
      G.BP|G|TR(NFG|G|TR) = 0 ;
   END ;
   G.INT(NLG|G|TR) = G.INT(G|G|TR) ;
   NES = NES + 1 ;
   IF NES > LNES THEN LNES = NES ;
   KLC(J) = KLC(J) + 1 ;
   KL(J,KLC(J)) = NLG|G|TR ;
   CALL CHECK(G.INT(G|G|TR)) ;
   GO TO END1 ;
   END ;
ELSE /* CHECK FOR CONDITION 2: */
   IF TAND = INTP(IJ) THEN
      /* CONDITION 2 */
      M(IJ) = '1'6 ;
   END ;
END OVERJ ;

TBTMULT:
   /* DO TERM BY TERM MULTIPLY */
   DO J = 1 TO NUM :
      TAND = INTP(IJ) & SUBSTR(G.INT(G|G|TR),
                     NVP(POSP(IJ)),NLEV(POSP(IJ))) ;
      IF TAND = ZERO5 THEN GO TO ENDMULT ;
      TAI|G|TR = G.INT(G|G|TR) ;
      SUBSTR(TAI|G|TR,NVP(POSP(IJ)),NLEV(POSP(IJ))) = TAND ;
      /* CHECK TO SEE IF ANY OF THE PREVIOUS TERMS GENERATED OUT OF THE JTH INTERVAL COVERS THIS NEW INTERVAL */
      DO IKL = 1 TO KLC(J) :
         IF (G.INT(KLC(J),IKL)) & TAI|G|TR = TAI|G|TR THEN GO TO ENDMULT ;
      END ;
      IF ~MI(J) THEN /* CHECK THIS CONDITION */
         INTERVAL TO SEE IF IT IS COVERED BY A PREVIOUS CONDITION */
         DO IKL = 1 TO KBC(J) :
            IF (G.INT(KBC(J),IKL)) & TAI|G|TR = TAI|G|TR THEN GO TO ENDMULT ;
         END ;
      /* PUT INTERVAL INTO NEW G(I(E) */
      CALL GETG(NLG|G|TR) ;
IF NGPTR = 0 THEN
  DO:
    NGPTR = NLGPTR - 1;
    G.BPTR(NGPTR) = 0;
  END:
  G.INT(INLGPTTR) = TAINT;
  NES = NES + 1;
  IF NES > LNES THEN LNES = NES;
  IF STRACE THEN
    PUT SKIP(2) EDIT('NEW INTERVAL','1(A);
    CALL PICKUP(TAINT); END:

  IF MIJ THEN
    DO:
      /* CHECK TO SEE IF THIS NEW INTERVAL 
        COVERS ANY PREVIOUS TERMS GENERATED 
        OUT OF THE JTH INTERVAL */
      CALL CHECKTAINT();
      /* PUT IN KL POINTER TO THIS ONE 
        COULD COVER LATER INTERVALS */
      KLC(J) = KLC(J) + 1;
      KL(J,KLC(J)) = NLGPTR;
      /* PUT IN KE POINTER FOR THIS ONE 
        AS IT CAN BE COVERED LATER */
      KEC(J) = KEC(J) + 1;
      KE(J,KEC(J)) = NLGPTR;
    END:
    ELSE DO: /* CHECK COND4 COVERAGE */
      CALL CHKTAINT();
      /* PUT IN KB POINTER AS THIS 
        INTERVAL IS A COND4 INTERVAL */
      KBC(J) = KBC(J) + 1;
      KB(J,KBC(J)) = NLGPTR;
    END:
  END:
ENDMULT:
END TBMULT:

CPTR = G.FPTR(GPTR); /* "INCREMENT" "I" */
IF CPTR = 0 THEN GO TO ZEROM; /* TEST FOR END 
  OF LOOP */
CALL RETURN(FCPTR,LGPTR):
CPTR = NGPTR; /* UPDATE POINTERS */
LCPTR = NLGPTR;
G.FPTR(LGPTR) = 0;
IF STRACE THEN CALL PRINTST:
IF OST THEN PUT SKIP(2) EDIT('FINISHED MULT. FOR FOE(10)');
  'STAR SIZE =', NES,
(*/LARGEST INTERMEDIATE STAR SIZE = *,LNES)
(A,F(4),A,X(18),A,F(5),X(4),A,F(5));
IF (LNES > MAXSTAR) & (TO < NODM)
THEN CALL TRIMSTAR;
END MPROD;

IF LNES > LNEAL THEN LNEAL = LNES ;
IF NES > LSS THEN LSS = NES ;
IF NUMTRIM > 0 THEN NUMTRIM = NUMTRIM + 1 ;
RETURN ;

312 4 0 TEQF: DO I=1 TO NV:
313 4 1 KRC(I)=INDEX(SUBSTR(EK(I),I1L),NVPI(I),NLPIV(I)),'1''B' - 1:
314 4 1 END:
315 4 0 PUT SKIP(3) EDIT
('*** DATA ERROR: TRUE EVENT = FALSE EVENT = ','KBC)
(A,NVIF(61));
316 4 0 IF FMIX THEN GO TO NEXTV:
317 4 0 CALL RETUN(FGPTR,LPTR) ;
318 4 0 GO TO NEXTV:
319 4 0 END STAR ;

/* ************************************************************************
 **************************************************************************/
320 3 0 TRIMSTAR: PROCEDURE REORDER :
/* THIS PROCEDURE TRIMS THE CURRENT STAR SIZE DOWN TO
CUTSTAR SIZE BY CHOOSING OUT OF THE NES ELEMENTS OF THE STAR
THE CUTSTAR ELEMENTS WHICH COVER THE MOST EVENTS FROM THE
CURRENT FNDMQ. */
321 4 0 DECLARE (CPTR,CNLT) CUTSTAR FIXED BINARY(15),
(MINPTR,MINT,NLIT,GPTR,1)
FIXED BINARY(15) STATIC ;
(MINW,WT) FIXED BINARY(31) STATIC,
CWT(CUTSTAR) FIXED BIN(31);
/* DEFINITIONS OF VARIABLES DECLARED IN THIS PROCEDURE:
CNLT - (CUT STAR NUMBER OF LITERALS) ARRAY CONTAINING
NUMBER OF LITERALS IN EACH OF THE CUT STAR INTERVALS
CPTR - (CUT STAR POINTER) ARRAY OF POINTERS TO THE
ELEMENTS IN THE CUT STAR
CWT - (CUT WEIGHTS) ARRAY CONTAINING THE WEIGHTS OF
THE ELEMENTS IN THE CUT STAR
GPTR - (G POINTER) POINTER USED FOR RUNNING THROUGH
*/
THE ORIGINAL STAR ELEMENTS

I - INDEX USED FOR INDEXING THROUGH CUT STAR
MIN - INDEX OF ARRAYS CPTR AND CWT WHICH POINTS TO THE ELEMENT OF THE CUT STAR WITH THE MINIMUM WEIGHT
MINPTR - POINTER TO THE ELEMENT OF THE CUT STAR WITH MINIMUM WEIGHT
MINWT - MINIMUM WEIGHT IN CUT STAR
NLIT - NUMBER OF LITERALS IN AN INTERVAL
WT - WEIGHT

GPRTR = FQPRTR; /* SET POINTER TO FIRST ELEMENT OF STAR */
MINWT = 50000;
DO I = 1 TO CUTSTAR; /* GET INITIAL ELEMENTS FOR CUT */
   CPTR(I) = GPRTR; /* STAR BY TAKING FIRST CUTSTAR */
   CALL LCORDER(GPRTR,W)); /* ELEMENTS OF STAR */
   CWT(I) = WT;
   NLIT=NLIT+1(GPTR);
   CNLT(I) = NLIT;
   IF WT > MINWT THEN GO TO INCR_PTR ;
   IF WT = MINWT THEN
      IF NLIT <= CNLT(MINI) THEN GO TO INCR_PTR ;
   MINWT = WT ;
   MINI = I ;
   INCR_PTR:
      GPRTR = GPRTR(GPTR) ;
   END :
   CALL LCORDER(GPRTR,W); /* REMOVE THIS ELEMENT FROM OLD STAR */
   IF WT < MINWT THEN /* REMOVE FROM OLD_STAR */
      GO TO REMOVE_FROM_OLD_STAR ;
   NLIT=NLIT+1(GPTR);
   IF WT = MINWT THEN IF NLIT > CNLT(MINI) THEN
      GO TO REMOVE FROM_OLD_STAR ;
   /* ADD THIS L TO CUT STAR AND REMOVE ELEMENT MINI */
   MINPTR = CPTR(MINI) ;
   CNLT(MINI) = NLIT ;
   CPTR(MINI) = GPRTR ;
   CWT(MINI) = WT ;
   IF GPRTR(GPTR) = 0 THEN /* FIND MINIMUM WEIGHT */
      DO :
         MINWT = 50000 ;
         DO I = 1 TO CUTSTAR ;
            IF CWT(I) > MINWT THEN GO TO NO_EXCHANGE ;
         IF CWT(I) = MINWT THEN
            IF CNLT(MINI) >= CNLT(I) THEN
               GO TO NO_EXCHANGE ;
         MINWT = CWT(I) ;
MINI = I ;

NO_EXCHANGE; END ;
END : 
GO TO INCR_1 ;
REMOVE_FROM_OLD_STAR ;
MINPTR = GPTR ; 
INCR_1 : 
GPTR = 6.FPTR(GPTR) ;
CALL REMOVE GMINPTR ; 
IF GPTR = = 0 THEN GO TO ELIM SMALL ; /* END OF STAR? */
NO = CUTF STAR ; /* YES, CLEANUP AND RETURN */
IF OST I SPACE THEN PUT SKIP(2) LIST 
('**** STAR REDUCED TO CUTF STAR SIZE ****');
IF TRACE THEN CALL PRTSTAR ;
NUMTRIM = NUMTRIM + 1 ;
RETURN ;
END TRIMSTAR :

PROCEDURE(GPTR) RETURNS(FIXED BINARY(15)) REORDER:
/* THIS PROCEDURE DETERMINES ALIT, THE NUMBER OF LITERALS IN 
THE INTERVAL OF G(I) POINTED TO BY GPTR. */
DECLARE (GPTR,ALIT) FIXED BINARY(15) ;
LIT FIXED BINARY(13) STATIC ;
ALIT = 0 ;
DO LIT = 1 TO NW ;
IF SUBSTRG.INT(GPTR),NVP(LIT),NLEV(LIT))
== SUBSTRNG(ONES,1,NLEV(LIT))) THEN ALIT=ALIT+1;
END :
RETURN(ALIT);
END NUMLIT :

PROCEDURE(L,NFIL) REORDER :
/* THIS PROCEDURE DETERMINES NFIL, THE NUMBER OF 
ELEMENTS OF THE CURRENT FINDMQ THAT ARE COVERED BY THE INTERVAL 
IN G(I) POINTED TO BY THE POINTER L */
DECLARE (II STATIC,L) FIXED BINARY(15) ,
NFIL FIXED BINARY(15) ;
NFIL = 0:
DO I=1 TO NE(K):
   IF FINOMQ(I) E EINCPXL(I,L) THEN NFIL = NFIL + 1:
END:
RETURN:
END LCOVER:

/* *****************************************************************
** THIS PROCEDURE CHOOSES FROM THE CURRENT STAR THE COMPLEX
** (FACTOR OR INTERVAL) WHICH BEST SATISFIES THE
** CRITERIA SPECIFIED IN CLIST. FOR EACH CRITERION,
** A VALUE (CRIVAL) IS DETERMINED FOR EVERY
** COMPLEX IN THE STAR. THOSE COMPLEXES WHOSE CRIVAL LIES
** WITHIN A RANGE OF THE MINIMUM CRIVAL ARE CONSIDERED
** FOR THE NEXT CRITERION, THE REMAINING ONES BEING DELETED.
** THIS RANGE IS DETERMINED AS A PERCENTAGE (GIVEN IN TLIST)
** OF THE OVERALL RANGE OF CRIVALS. WHENEVER A SINGLE
** COMPLEX REMAINS, THE ROUTINE RETUNS IMMEDIATELY WITH
** ITS LO-POINTER. OTHERWISE, WHEN ALL CRITERIA
** ARE PROCESSED, THE FIRST COMPLEX WITH MINIMUM CRIVAL
** IS CHOOSEN. THE COMPLEX IS ADDED TO THE CURRENT H2-LIST
** AND ALL APPROPRIATE FLAGS & COUNTS ARE UPDATED. */

DECLARE (ICRIT,CRITX,LPTR,NUMCPXL,A,B,LASTPOS)
   FIXED BINARY(5),
   (UBOUND,MAYVAL,MINVAL,V) DEC FLOAT;
NUMCPXL,CVNSSTAR=0; LPTR=GPTR;
DO WHILE (LPTR=0): /* COUNT # COMPLEXES IN STAR AND
   SET FINOSTAR TO *0*B FOR ALL
   EVENTS IN STAR */
   NUMCPXL=NUMCPXL+1:
   DO I=1 TO NE(K):
      IF FINOSTAR(I) E EINCPXL(I,LPTR) THEN DO:
         FINOSTAR(I)=*0*B; CVNSSTAR=CVNSSTAR+1:
      END;
   END:
   LPTR=GP(LPTR);
   END:
IF LTRACE THEN DO:
   PUT PAGE EDIT ('STAR FOR TRUE EVENT',IIL,' IN CLASS',K)
(A,F(4),A,F(3));
LPTR=FGPTR;
DO WHILE (LPTR=0):
    CALL PCOVERIG(INLPTR);
    CALL SKIP LIST(*CRIT VALUE*);
    DO ICritt=1 TO NCritt;
       CRITT=CLIST(ICritt);
       CALL SKIP EDIT(CRITT,ABS(CRITVAL(CRITT))
       [F(3),F(10),2]);
    END;
    LPTR=GF PTR(LPTR);
END:

PUT SKIP(2) EDIT ('LARGEST INTERMEDIATE STAR SIZE WAS',
LNES)(A,F(5));
END:

CRITLOOP:
    DO ICritt=1 TO NCritt;
       MINVAL=1.6601;
       IF NUMCPLX=1 THEN GOTO CHOOSE;
       MAXVAL=MINVAL;
       CRITT=CLIST(ICritt);
       LPTR=FGPTR;
       QQLOOP: DO WHILE (LPTR=0):
          /* CALCULATE CRITVAL FOR GIVEN LQ */
          V, G.VAL(LPTR)=CRITVAL(CRITT);
          IF MAXVAL(V) THEN MAXVAL=W;
          IF MINVAL>V THEN MINVAL=V;
          LPTR=GF PTR(LPTR);
       END QQLOOP;
       UBOUND=MINVAL+TLIST(ICRITT)*MAXVAL-MINVAL;
       LPTR=FGPTR;
       DO WHILE (LPTR=0):
          IF G.VAL(LPTR) > UBOUND THEN DO:
             /* DELETE LQ'S WHICH DON'T MEET CRITERIA */
             I=G.F PTR(LPTR);
             CALL REMOVE(LPTR);
             LPTR=I;
          END;
          NUMCPLX=NUMCPLX-1;
       ELSE LPTR=GF PTR(LPTR);
       END:
    END CRIT LOOP:

/* CHOOSE COMPLEX WITH MINIMUM CRITVAL */
CHOOSE: LPTR=FGPTR;
DO I=1 TO NUMCPLX WHILE (MINVAL < G.VAL(LQ));
LO=G.FPTR(LQ); END;

IF LQTR. THEN DO;
   IF LQSTOP THEN DO:
      /* DETERMINE LQ-STAR FROM CURRENT LQ */
      I.E. OPTIMIZE CHOOSEN COVER */
      TAINT=ZERO;
      DO I=1 TO N(E(K));
      IF FINOMQ(I) & EINCPLX(I,LQ) THEN
         TAINT=TAINT | E(K,I);
      END;
      DO I=1 TO NV;
      IF SUBSTR(G.INT(LQ),NP(A),NLEV(I))
         = SUBSTRONES,L,NLEV(I) THEN
         SUBSTRTAINT,NP(A),NLEV(I)=ONES;
      END;
      IF TYPCOV(I) THEN DO:
         /* CONVERT FACTORS INTO INTERVALS */
         LASTPOS=NVP(I)+NLEV(I)-1;
         DO =NVP(I) TO LASTPOS
      WHILE (-SUBSTRTAINT,A,I));
      END; /* A = POS. OF 1ST '1' IN VARIABLE */
      DO B=LASTPOS TO NVP(I) BY -1
      WHILE (+SUBSTRTAINT,B,I));
      END; /* B = POS. OF LAST '1' IN VARIABLE */
      SUBSTRTAINT,A,B-A+1=ONES;
   END;
G.INT(LQ)=TAINT;
END;

/* ADD CHOOSEN LQ TO CURRENT MQ-LIST */
MQPTR*MQPTR+1;
IF MQPTR > NMQ THEN DO;
   PUT SKIP(3) LIST('### MQ SPACE EXCEEDED ###');
   CALL RETURN(LQ,FQPTR);
   GOTO NEWDATA;
END;
MQ(MQPTR) = LQ, BY NAME;
IF NQTRIM > 0 THEN MQ.TRIM(MQPTR) = '#';
ELSE MQ.TRIM(MQPTR) = ' *';

/* RESET PROPER ELEMENTS OF FINOMQ & COUNT
   NO. NEW EVENTS COVERED */
NQF(LQ)=0; CONCVD=0;
DO I=1 TO NE(K):
   IF FINDMQ(I) & EINCPLX(I,LO) THEN DO:
      FINDMQ(I)=0.0; NFILQ=NFILQ+1;
   END:
   IF EINCPLX(I,LO) THEN DO:
      COMCOVD=COMCOVD+1;
   END:
   END:
   NFILQ=NFILQ+NFILQ;
   TOTCVD(ISPL)=NFILQ;
   NEWCOV(ISPL)=NFILQ;
   NUMCOV(ISPL)=COMCOVD;
   ISPL=ISPL+1;
IF LITRACE THEN DO:
   PUT SKIP(2) EDIT('THIS LO COVERS*,NFILQ,
      'NEW EVENTS AND A TOTAL OF',NFILQ,
      'EVENTS ARE NOT COVERED IN MQ*),
      ('A,F(4),A,F(4),A');
   END:
   IF -QLQT < NUMTRIM THEN PUT SKIP(2) EDIT
      (** STAR WAS TRIMMED*,NUMTRIM,' TIMES ***)
      ('A,F(4),A');
   END:
   CALL RETURN(FGPTR,TLPTR);
RETURN;
PROCEDURE(CRITX) RETURNS(DEC,FLOAT) REORDER;
DECLARE (1,CRTX) FIXED BINARY(15), CRTX(F) LABEL,
   VI DEC FLOAT STATIC, STRING BIT(NLMAX) VARYING;
TAINT=G.INT(LPTR);
V, VI = 0.0;
GO TO CRIT(CRTX);
CRIT(1): /* MAXIMIZE # OF EVENTS NOT COVERED BY MQ */
   DO I=1 TO NE(K):
      IF FINDMQ(I) & EINCPLX(I,LPTR) THEN V=V+1.0;
   END:
RETURN(-V);
CRIT(2): /* MINIMIZE # VARIABLES (LITERALS) */
   RETURN(FLOAT(INUMLIT(LPTR))));
503 50  CRIT(3): /* MINIMIZE SUM(COST-PER-VARIABLE) */
      DO I=1 TO NVI
         IF SUBSTR(INT,NVP(I)+NLEV(I))
             THEN V=V+Z(I);
      END;
      RETURN(V);

507 50  CRIT(4): /* MINIMIZE GENERALIZATION COEFFICIENT */
      V = 1.: DO I=1 TO NV; /* FIND TOTAL # CELLS IN COMPLEX */
      VL = RANGE(I);
      IF VL<0. THEN VI=NLEV(I);
      V = V*VL;
      END;
      RETURN(V/V1);

518 50  CRIT(5): /* MAXIMIZE SUM(WEIGHTS OF EVENTS NOT IN HQ) */
      DO I=1 TO NE(K);
         IF FINDMOI(I) & EINCPLX(I,LPTR) THEN V=V+MO(K,I);
      END;
      RETURN(-V);

522 50  CRIT(6): /* MINIMIZE SUM(RANGES OF VARS IN COMPLEX) */
      CRIT(7): /* MINIMIZE SUM(VARIANCES OF RANGES OF VARS IN COMPLEX */
      DO I=1 TO NV;
         IF CRIT=6 THEN V=V+RANGE(I);
         ELSE V=V+VARIANCE(I);
      END;
      RETURN(V);

527 50  RANGE: PROCEDURE(IVAR) RETURNS(DEC FLOAT) REORDER:
      DECLARE (IND,IVAR) FIXED BINARY(15), RG DEC FLOAT;
      RG = 0.;
      STRING = SUBSTR(INT,NVP(I)+NLEV(IVAR));
      IF STRING=SUBSTRONES(I,NLEV(IVAR)) THEN RETURN(RG);
      CHKIND: IND=INDEX(STRING,'1'B1);
      IF IND<0 THEN RETURN(RG);
      RG=RG+1.: GO TO CHKIND;
      END RANGE;

538 50  VARIANCE: PROCEDURE(IVAR) RETURNS(DEC FLOAT) REORDER:
      DECLARE (IVAR,J,NLEV,LEV1:NLEV(IVAR))) FIXED BINARY(15),
(AVG,VCE) DEC FLOAT STATIC;

540   6   0   VCE = 0.1;
541   6   0   STRING = SUBSTR(TAMT,NVP+IVAR),NLEV+IVAR);
542   6   0   IF STRING=SUBSTR(ONES-1,NLEV+IVAR) THEN RETURN(VCE);
543   6   0   LEV = 0;
544   6   0   NLEV = 0;
545   6   0   DO I=1 TO NLEV+IVAR;
546   6   1   IF SUBSTR(STRING,I,NLEV+IVAR) THEN DO;
547   6   2   NLEV=LEV+1;
548   6   2   LEV(I)=I;
549   6   2   END;
550   6   1   END;
551   6   0   AVG=SUM(LEV)/NLEV;
552   6   0   DO I=1 TO NLEV+IVAR;
553   6   1   IF LEV(I) == 0 THEN VCE=VCE+ABS(AVG-LEV(I));
554   6   1   END;
555   6   0   RETURN(VCE/NLEV);
556   6   0   END VARIANCE;
557   5   0   END CRITYAL;

/="/-----------------------------------------------------------------------/"
558   4   0   END BESTLQ;

/="/*******************************************************************/
559   3   0   PRTRSTAR: PROCEDURE REORDER:
560   4   0   DECLARE GPTR FIXED BINARY(15) STATIC;
561   4   0   GPTR = FPTR ;
562   4   0   DO WHILE (GPTR != 0);
563   4   1   CALL PROGRESS(NP,INT(GPTR));
564   4   1   GPTR = G.FPTR(GPTR);
565   4   1   END;
566   4   0   RETURN ;
567   4   0   END PRTRSTAR ;

/="/****************************************************************+++*/
568  3  0 |PRTLQ: PROCEDURE REORDER ;
   /* THIS PROCEDURE PRINTS OUT ABBREVIATED INFORMATION
      ABOUT THE NEWLY OBTAINED STAR AND THE ASSOCIATED LQ. */
   PUT SKIP(3) EDIT(REPEAT('-',100))(A);
   PUT SKIP(1) EDIT
      ('STAR*' ,MQPTR,' COVERING EVENT',ILL, 'HAS' ,NES,
     ' COMPLEXES: LARGEST INTERMEDIATE STAR SIZE WAS',LNES)
      (A,F(4),A,F(4),A,F(5),A,F(5)) ;
   IF NUMTRIM > 0 THEN PUT SKIP(2) EDIT
      ('** NUMBER OF TIMES STAR WAS TRIMMED = ',NUMTRIM,
      ' ***)' (A,F(4),A) ;
   IF SAVELO THEN PUT FILE(COVER) LIST(MQ,INT(MQPTR));
   CALL PROCOVER(MQ,INT(MQPTR));
   PUT SKIP(2) EDIT
      ('THIS LQ COVERS',NFLQ,'NEW EVENTS AND A TOTAL OF',NFLQ,
      ' EVENTS ARE NOW COVERED IN MQ')
      (A,F(4),A,F(4),A) ;
   RETURN ;
   END PRTLQ ;

   /***************************************************************
   **                                                                         **
   ** PROCEDURE REORDER ;                                                     **
   **                                                                         **
   ** THIS PROC READS IN THE EVENTS ASSOCIATED WITH THE CLASSES               **
   ** E(0,*),...,E(MAXCL,* ) AND CONVERTS THEM TO BI-INARY STRINGS.           **
   ** input events are assumed to be in vector form. */
   DECLARE (I,J,K) FIXED BINARY(15) ;
   DO K=0 TO MAXCL:
   PUT SKIP(2) EDIT('CLASS F(',K,'')'(A,F(2),A);
   DO I=1 TO NE(I);
   GET LIST(POSP);
   PUT SKIP EDIT(EVENT NO.',I,'**',POSP
     (X(5),A,F(4),A,(NV)F(6)) ;
   E(K,I)=ZEROS;
   DO J=1 TO NV;
   IF POSP(J)<0 OR POSP(J)>NLEV(J)-1 THEN DO;
   PUT SKIP(2) EDIT('** VARIABLE',
     ' INCORRECTLY SPECIFIED',
     ' IN ABOVE EVENT ***)'(A,F(3),A,A) ;
   GO TO NEWDATA;
   END:
SUBSTR(E(K, I), NVP(J) + POSP(J), 1) = '1'B;

---

PROCEDURE REORDER:

/* THIS PROC READS IN THE EVENTS ASSOCIATED WITH THE CLASSES E(0, T), ..., E(MAXCL, T) AND CONVERTS THEM TO BI-UNARY STRINGS. INPUT EVENTS ARE ASSUMED TO BE IN GAMMA FORMAT. */

DECLARE II, K, BVAL FIXED BINARY(15), J FIXED BINARY(31);

DO K = 0 TO MAXCL:

   PUT SKIP(2) EDIT('CLASS F('T, K, 'T')'(A,F(2), A);

   DO I = 1 TO NE(K):

      GET LIST(J);

      PJT SKIP EDIT('EVENT NO.*', I, 'M' = J)
      (X(15), A, F(6), A, F(6));

      E(K, I) = ZEROS;

      DO II = M+1 BY 1 TO I

         BVAL = MOD(J, NLEV(II));

         J = J / NLEV(II);

      END:

      SUBSTR(E(K, I), NPI(II) + BVAL, 1) = '1'B;

   END:

END REORDER:

---

PROCEDURE (E1, EO, IM, PH, NM) REORDER:

/* THIS PROCEDURE PERFORMS THE OPERATION E1 MIX (EO COMPLEMENT) THE RESULT IS STORED IN AN ABBREVIATED FORM IN ARRAYS IM AND PH.

IM[I] CONTAINS THE BOUND PAIR OF THE LITERAL.
PH[I] CONTAINS THE POSITION OF THE BOUND PAIR I IN THE
INTERVAL (ALL OTHER POSITIONS HAVE MAXIMUM UPPER BOUND
AND MINIMUM LOWER BOUND).
NM IS THE NUMBER OF INTERVALS. */

DECLARE (E1, EO) BIT(*), IM(*) BIT(*), PH(*) BIT(*), NM FIXED BINARY(15),
((IP, J, K, BITPOS) STATIC), NM) FIXED BINARY(15).
(PARTEO, PARTEI) BIT(NLMAX) ALIGNED;

NM = 0;
DO IP = 1 TO NV;
  PARTEO = SUBSTR EO(NV,IP), NLEV(IP) ;
  PARTEI = SUBSTR EL(NV,IP), NLEV(IP) ;
  IF PARTEO = PARTEI THEN GO TO END THIS POSITION;

NM = NM + 1;
PH(NN) = IP ;
  IF TYPCOV(IP) THEN DO;
    IM(NM) = SUBSTR -PARTEO1, NLEV(IP) ;
    GO TO END THIS POSITION;
  END;
  BITPOS = INDEX(PARTEI, "1") ;
  IM(NM) = "0" Bi;
  DO J = BITPOS + 1 TO NLEV(IP) WHILE (SUBSTR(PARTEJ, J)) ;
  DO K = BITPOS + 1 TO 1 BY -1 WHILE (SUBSTR(PARTEO, K)) ;
  SUBSTR (IM(NM)), K+1, J-K-1 = "0"ES;

END THIS POSITION:
END;
RETURN ;

END MIX ;

/* ===============================================================
  CHECKER: PROCEDURE (BP) REORDER :
  /* THIS PROCEDURE CHECKS EARLIER INTERVALS IN GIE FOR
   POSSIBLE COVERAGE BY THE INTERVAL REPRESENTED BY THE BOUND
   PAIR OF. ANY PREVIOUS INTERVALS COVERED ARE THROWN OUT
   OF GIE AND ITS ENTRY IS DELETED FROM THE KE LIST. */
  DECLARE BP BIT(*) ALIGNED,
        (THROWE, THROWL) BIT(1) ALIGNED STATIC,
        (IL, IKL, LP) FIXED BINARY(15) STATIC ;

THROWE = '0' Bi;
THROWL = '0' Bi;
DO IK = 1 TO KE(J) ;
P = KE(J, IK) ;
IF (BP = G, INT(P)) = G, INT(P) THEN
  /* NEW INTERVAL COVERS OLD ONE: */
  THROW OUT OLD ONE.
  DO ;
    NES = NES - 1 ;
  CALL REMOVEON(P) ;
P = KE(J, IK) ;
/* SEE IF THIS POINTER IS IN KL LIST */
DO IKL = 1 TO KLC(J) :
   IF KL(IJK) = I THEN /* YES, REMOVE IT */
      DO :
         THROWL = '1'B;
         KL(J,IKL) = 0 ;
         KLC(J) = KLC(J) - 1 ;
         GO TO REMOVEE :
      END :
   END |
/* REMOVE POINTER FROM KE LIST */
REMOVEE: KEC(J) = KEC(J) - 1 ;
KE(IKE) = 0 ;
THROWE = '1'B ;
END :
CALL CKBE(BP) :
IF THROWE THEN /* COLLAPSE KE LIST */
   DO :
      I = 0 ;
      DO IKE = 1 TO KEC(J) :
         INCRE: I = I + 1 ;
         IF KE(IKE) = 0 THEN GO TO INCRE ;
         KE(IKE) = KE(IKE) :
      END :
   END |
IF THROWL THEN /* COLLAPSE KL LIST */
   DO :
      I = 0 ;
      DO IKL = 1 TO KLC(J) :
         INCRL: I = I + 1 ;
         IF KL(IJ,K) = 0 THEN GO TO INCRL ;
         KL(IK) = KL(IK) :
      END :
   END :
RETURN :
END CHECKE :
/* ****************************************************
   **************************************************** /
CKBE: PROCEDURE(BP) REORDER ;
   /* THIS PROCEDURE CHECKS EARLIER CONDITION 4 INTERVALS
   IN GIE) FOR POSSIBLE COVERAGE BY THE INTERVAL REPRESENTED
BY THE INTERVAL BP. ANY PREVIOUS INTERVALS SO COVERED
ARE THROWN OUT OF G(E) AND ITS ENTRY IS DELETED FROM
THE KB LIST. */

DECLARE BP BIT(*) ALIGNED,
    THROBP Bit(1) ALIGNED STATIC;

DECLARE KB(2) Bit(1) FIXED BINARY(15) STATIC;

THROBP = 'Q'B;
DO IKB = 1 TO KB(J);
    P = KB(J,IKB);
    IF (BP & G.INT(P)) = G.INT(P) THEN /* INPUT INTERVAL
        COVERS OLD ONE; THROW OUT OLD ONE. */
        DO ;
            NES = NES - 1:
            CALL REMOVEP(NP);
            /* REMOVE POINTER FROM KB LIST */
            KB(J) = KB(J) - 1;
            KB(J,IKB) = 0;
            THROBP = '1'B;
        END;
    END ;

IF THROBP THEN /* COLLAPSE KB LIST */
    DO ;
        I = 0;
        DO IKB = 1 TO KB(J);
            INCR: I = I + 1;
        IF KB(J,IKB) = 0 THEN GO TO INCR;
        KB(J,IKB) = KB(J,I);
    END;
    RETURN;
END CKBE:

REMOVEP: PROCEDURE(IP) REORDER ;
/* THIS PROCEDURE REMOVES THE ELEMENT POINTED TO BY P
FROM THE STAR AND RETURNS IT TO THE FREE LIST PART OF G(E).
ENTRY REMOVEP IS USED IF REMOVAL IS FROM A STAR BEING
GENERATED; IN THIS CASE IT IS ASSUMED THAT THE ELEMENT
BEING REMOVED IS NEVER THE LAST ONE. */

DECLARE P FIXED BINARY(15) ;
699  4 0  IF P = NFGPTR THEN NFGPTR = G.FPTR(P) ;
700  4 0  SETPTRS:
    IF G.FPTR(P) == 0 THEN
      G.BPTR(G.FPTR(P)) = G.BPTR(P) ;
    IF G.BPTR(P) == 0 THEN
      G.FPTR(G.BPTR(P)) = G.FPTR(P) ;
702  4 0  CALL RETURNING(P, P) ;
703  4 0  RETURN ;
704  4 0  REMOVEG: ENTRY(P) :
    /* THIS ENTRY IS USED IF REMOVAL IS FROM AN OLD STAR */
705  4 0  IF P = FGPTR THEN FGPTR = G.FPTR(P) ;
706  4 0  IF P = LGPTR THEN LGPTR = G.BPTR(P) ;
707  4 0  GO TO SETPTRS ;
708  4 0  END REMOVEG ;
/* ===================================================================
   */
709  3 0  GETG:  PROCEDURE(NEXT) REORDER ;
    /* THIS PROCEDURE GETS THE NEXT ELEMENT OF G FROM THE
UNUSED LIST. NEXT IS SET TO THE ELEMENT SO OBTAINED. */
710  4 0  DECLARE NEXT FIXED BINARY(15) ;
711  4 0  IF NEXTFG = 0 THEN
      DO :
    712  4 1  PUT SKIP(3) LIST
      (**** NO MORE SPACE IN G ****) ;
    713  4 1  CALL RETURNING(NFGPTR, LGPTR) ;
    714  4 1  CALL RETURNING(FG PTR, LGPTR) ; GO TO NEWDATA ;
    716  4 1  END : 
    717  4 0  NEXT = NEXTFG : /* SET POINTER TO NEXT AVAILABLE ELEMENT */
    718  4 0  NEXTFG = G.FPTR(NEXTFG) ; /* MOVE UP NEXTFG POINTER */
    719  4 0  IF NEXTFG = 0 THEN LASTFG = 0 ; /* CHECK ENDING CONDITION */
    720  4 0  IF STRACE THEN PUT SKIP(1) EDIT
      (GETG*, NEXT) (A,F(15)) ;
    721  4 0  RETURN ;
722  4 0  END GETG ;
/* ===================================================================
   */
RETURNING: PROCEDURE(FIRST, LAST) REORDER:

/* THIS PROCEDURE RETURNS A LIST OF G ELEMENTS TO THE
UNUSED LIST. FIRST POINTS TO THE FIRST ELEMENT AND LAST
POINTS TO THE LAST ELEMENT OF THE LIST TO BE SO RETURNED. */

DECLARE (FIRST, LAST) FIXED BINARY(15);

IF STRACE THEN PUT SKIP(1) EDIT
('RETURNING', FIRST, LAST) (A, (2) F(15)) ;

G.BPTR(FIRST) = LASTFG ; /* SET FIRST BACK POINTER TO POINT
TO THE LAST FREE G ELEMENT */

G.FPTR(LAST) = 0 ; /* MAKE SURE LAST FORWARD POINTER IS
NULL */

IF LASTFG = 0 THEN G.FPTR(LASTFG) = FIRST ; /* SET PREVIOUS
LAST FORWARD POINTER */

LASTFG = LAST ; /* MOVE UP LASTFG POINTER */

IF NEXTFG = 0 THEN NEXTFG = FIRST ;

RETURN;

END RETURNING :

END EINCPLX :

PROCEDURE(I1, GPTR) RETURNS(BIT(I1)) REORDER:

/* THIS PROCEDURE TESTS THE TRUE EVENT E(K, I1) FOR
INCLUSION IN THE INTERVAL OF G(E) POINTED TO BY GPTR.
INCLUSION RESULTS IN A RETURN OF '1' IF E IS RETURNED. */

DECLARE (I1, GPTR) FIXED BINARY(15);

RETURN((G.INT(GPTR) & E(K, I1)) = E(K, I1));

END EINCPLX:

PROCEDURE(COVRA) REORDER:

DECLARE COVR BIT(*) ALIGNED, STRING BIT(NLMAX) VAR,
(I, M, J, IND) FIXED DECIMAL(3),
NUMB CHAR(3) VARYING;

M=0;

PUT SKIP;
741 4 0  |  LINE='COMPLEX: *';
742 4 0  |  DO I=1 TO NV;
743 4 1  |    STRING=SUBSTR(COVV,NVP(I),NLEV(I));
744 4 1  |    IF STRING=SUBSTR(ONES,1,NLEV(I)) THEN GOTO NEXTV;
745 4 1  |    IF [ > 9 ] THEN NUMB=SUBSTR(CHAR(I),5);
746 4 1  |    ELSE NUMB=SUBSTR(CHAR(I),6);
747 4 1  |    LINE=LINE || ' (X || NUMB || *)';
748 4 1  |    CHKIND=IND=INDEX(STRING,'1'');
749 4 1  |    IF IND=0 THEN GOTO RPAREN;
750 4 1  |    J=IND-1;
751 4 1  |    IF J > 9 THEN NUMB=SUBSTR(CHAR(J),4);
752 4 1  |    ELSE NUMB=SUBSTR(CHAR(J),5);
753 4 1  |    LINE=LINE || NUMB;
754 4 1  |    SUBSTR(STRING,IND,11)=D*N;
755 4 1  |    GO TO CHKIND;
756 4 1  |   RPAREN=LINE=LINE || ' * ';
757 4 1  |   NEXTV: IF LENGTH(LINE) > 90 THEN DO:
758 4 2  |      PUT SKIP EDIT(LINE)(A);
759 4 2  |      LINE='
';
760 4 2  |      IF M=0 THEN DO:
761 4 3  |       M=M+1;
762 4 3  |       PUT EDIT(NUMCOV(ISPL),NEWCOV(ISPL),
763 4 3  |         INDEP(ISPL),TOTCOV(ISPL))((COL(100),
764 4 3  |            F(3),X(2),F(3),X(2),F(3),X(2),F(3));
765 4 2  |   END;
766 4 2  |   END;
767 4 0  |   IF LENGTH(LINE) > 10 THEN PUT SKIP EDIT(LINE)(A);
768 4 0  |   IF M=0 THEN PUT EDIT(NUMCOV(ISPL),NEWCOV(ISPL),
769 4 0  |      INDEP(ISPL),TOTCOV(ISPL))((COL(100),F(3),X(2),
770 4 0  |        F(3),X(2),F(3),X(2),F(3));
771 4 0  |   END PROCOVER;
772 3 0  | TRIPLE: PROCEDURE REORDER;
773 4 0  |   DCL (COMP(MOPTS)) ALIGNED
774 4 0  |      BITLEN INIT(MOPTS)*0'B';
775 4 0  |   DCL (I,J,L,TOTO(MOPTS)) FIXED BIN(15);
776 4 0  |   TOTO=0;
777 4 0  |   J=1;
778 4 0  |   DO I=OMPTR+1 TO MOPTR;
779 4 1  |      COMP(I)=M3_INT(I);
780 4 1  |      J=J+1;
781 4 1  |   END;
782 4 0  |   DO I=1 TO MOPTS; FINDOMQ='1'B;
783 4 1  |   DO J=1 TO MOPTS;
784 4 2  |      IF I<J THEN DO;
DO L = 1 TO NE(K):
  IF FINDMQ(L) & (COMP(I) & E(K,L)) = E(K,L)
    THEN DO:
      TOTO(I) = TOTO(I) + 1:
      FINDMQ(L) = '0'B1
    END:
  END:
END;
AQUAL/1 (AQ7)
User's Guide and Program Description

James Larsson and R. S. Michalski
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

National Science Foundation
Washington, D.C.

AQUAL/1 (AQ7) is a PL/1 program which synthesizes quasi-optimal formulas of the variable-valued logic system VL. By 'quasi-optimal formulas' we mean here disjunctive simple VL formulas, which are optimal or sub-optimal with regard to a user-specified optimality functional.

The basic application of the program is in the area of machine learning and inductive inference ('inductive learning'): from descriptions of objects with known class membership, the program infers optimal or sub-optimal descriptions of object classes. These descriptions are expressed as VL formulas and represent certain generalizations of inputted information. The program can also be used (at the appropriate setting of its parameters) for an efficient minimization of binary- or multi-valued switching functions with a large number of variables (e.g., 50-100 variables).