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, Urbana, Illinois, June 1975.

UIUCDCS-R-75-731

AQVAL/1 (AQ7)
User's Guide and Program Description

by

James Larson
R. S. Michalski

June 1975

UIUCDCS-R-75-731

AQVAL/1 (AQ7)

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,[1] 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, Celso Frazer, Kurt Hirchert and Richard Chilausky.

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

AQVAL/1: Part I

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 VL_1. 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 VL_1 given in paper[1] and therefore very limited explanation of the concepts of VL_1 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 paper[2]). In order to understand the last chapter, the reader should be acquainted with paper[2]. Bibliography, in addition to papers containing the background information for this paper (i.e., the definition[1] of the VL_1 and a description of synthesis algorithms[2]) 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 $\underline{A}^q$ 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 paper[20].

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: `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. 14, parameter F) which we will denote here as $F^0, F^1, \ldots, F^{m-1}$. (The assignment of symbols $F^0, F^1, \ldots, F^{m-1}$ to the event sets specified in the input stream depends on the OLIST 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 F^i, and value 0 for the remaining events, specified in F^j, $j \neq i$, $j=0,1,\ldots,m-1$:

  $$f_i(e) = \begin{cases} 1, & \text{if } e \in F^i \\ 0, & \text{if } e \in \mathbf{F} \setminus F^i \end{cases}$$

  where $\mathbf{F}$ is the union of all events specified in the program, i.e.

  $$\mathbf{F} = F^0 \cup F^1 \cup \ldots \cup F^{m-1}$$

  The output from the program consists of m DVL_1 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 $\mathbf{E}\setminus\mathbf{F}$, called *-events, where $\mathbf{E}$ is the universe of events). Formally, a $V(f_i)$ expression is defined as a DVL_1 expression corresponding to an optimized (with regard to functional A) cover of set F^i against the set $\mathbf{F}\setminus F^i$:

$$V(f_i) \leftrightarrow C_i = C(F^i | \mathbf{F}\setminus F^i) \quad i=0,1,\ldots,m-1$$

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 $\mathbf{F}$(i.e., event from the input list) satisfies exactly one formula (it follows from the assumption that sets F^i 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:

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

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

$$\vdots \qquad \vdots$$

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

$$\vdots \qquad \vdots$$

$$V(f_{m-1}) \leftrightarrow C_{m-1} = C(F^{m-1} \mid \bigcup_{j=0}^{m-2} C_j^{\cup})$$

(By $C_j^{\cup}$ 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:

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

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:

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

The output from the program is an optimized DVL_1 formula $V(f)$ expressing function f. Formula $V(f)$ is defined as:

$$V(f) = (m-1)(T_1^{m-1} \vee T_2^{m-1} \vee \ldots) \vee$$
$$(m-2)(T_1^{m-2} \vee T_2^{m-2} \vee \ldots) \vee$$
$$\vdots$$
$$1 \quad (T_1^1 \vee T_2^1 \vee \ldots)$$

where

T_i^{m-1}, $i=1,2,\ldots$ -- terms corresponding to complexes in the quasi-optimal (q-o) cover
$$C_{m-1} = C(F^{m-1} \mid \bigcup_{i=0}^{m-2} F_i)$$

T_i^{m-2}, $i=1,2,\ldots$ -- terms corresponding to complexes in the q-o cover
$$C_{m-2} = C(F^{m-2} \mid \bigcup_{i=0}^{m-3} F_i)$$

$\vdots$

T_i^1, $i=1,2,\ldots$ -- terms corresponding to complexes in the q-o cover
$$C_1 = C(F^1 \mid F^0)$$

Note that the total number of covers C_i is smaller by 1 than the number of sets F^i.

- INFORM

Example: `INFORM = 'VECTOR'`

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

If INFORM equals 'VECTOR', then events are read in as sequences of variable values. If INFORM equals 'GAMMA', then events are read in the form of event numbers $\gamma(e)$. If INFORM = 'VL', then events are expressed as a VL_1 formula.

- TITLE

Example: `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:

```
MAXSTAR = 150
CUTSTAR =  50
```

Possible values: MAXSTAR, CUTSTAR -- any positive integer smaller than NGE value (see next parameter)

These parameters are used to control the speed of the program. Small values of these parameters (e.g., MAXSTAR = 5, 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 MAXSTAR complexes, then it is reduced to CUTSTAR complexes, selected as the 'best' according to the assumed optimality functional (see CLIST and TLIST parameters).

---

*Implemented presently only in version AQ9 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 star 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: '0'B, '1'B

If LQST = '1'B each complex L^q selected from a star 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/1 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 F^i.

- 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: `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: `2 'FACTOR' 1 4`

  Possible values: NSPEC -- a positive integer ≤ 64 (this restriction can be easily removed)

  TYPE -- 'FACTOR' or 'INTERVAL'

  TYPELIST -- a list of NSPEC positive integers ≤ 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 VL_1 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, VL_1 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 5, 3, 2, 4`

Possible values: NV -- any positive integer ≤ 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 5, 1, 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 L^q. 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)*(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[12] and corresponding to them program criteria (in parentheses):

Criterion

#	Notation	Description of Attributes (Criteria)
1	t --	the number of terms in a formula (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 NMQ). The negative is used to have the minimum value for the criterion when the complex covers the maximum number of events.)
2	s --	the number of selectors in a formula (The number of selectors in a complex.)
3	z --	sum of the costs of the variables in a formula (z(v)) (The sum of the costs of the variables in a complex:

$$z' = \sum_{i \in I} z(i)$$

$z(i)$ -- 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^i 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)

$$\sum_{i \in I} \ell(xi)$$

I -- as in 3 above

$\ell(xi)$ -- number of constants in the reference of the selector for the variable xi.)

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,

$$\text{cost} = \sum_{i \in I} (MD_i)$$

$$MD_i = \sum_{r \in R_i} \frac{|avg_i - r|}{c(R_i)}$$

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

avg_i -- average value of the constants in the reference for variable xi 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 3   1 1 3   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 xi 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 xI is specified by Z(I)). Any variables for which no specification is given are assigned cost = 1. In the example given, variable x1 has cost 2, variable x4 has cost 1.2, and the other variables have cost = 1.

- W

  Example: W(0,2) = 1.3 W(1,1) = 2;

  Possible values: any real numbers

  This parameter is specified only if criterion 5 is specified. W(K,J) specifies the weight of J^{th} (J=1,2,3,...) event in the $(K+1)^{st}$ (K=0,1,...) 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 = 1.3, 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.

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):

- Unused core

  Example: AMOUNT OF UNUSED CORE = 4k

  This parameter specifies the amount of core memory which was estimated but not used.

- COMPLEX:

  Example: COMPLEX: (x1 = 0 2) (x3 = 1) (9, 6, 2, 12)

  This parameter specifies the actual complexes which were synthesized. Complexes of one formula (or in the case of 'VL' 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 F^i (i.e., event set associated with the current decision class)

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

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

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

- DELTA

Example: `DELTA for this set is 0`

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 NGE 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 AQVAL/1 (AQ7) 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 180k 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_1 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.

<u>GLD Description for Example #1</u>

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 x1 and x2 specify the row in which a particular event is to be placed, the variables x3 and x4 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 (i) in the squares correspond to the set F^i and to the terms of the DVL_1 formula with coefficient i. If the event sets are to be covered in IC or DC mode, then the numbers (i) 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^4, 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^4 and F^3. 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.

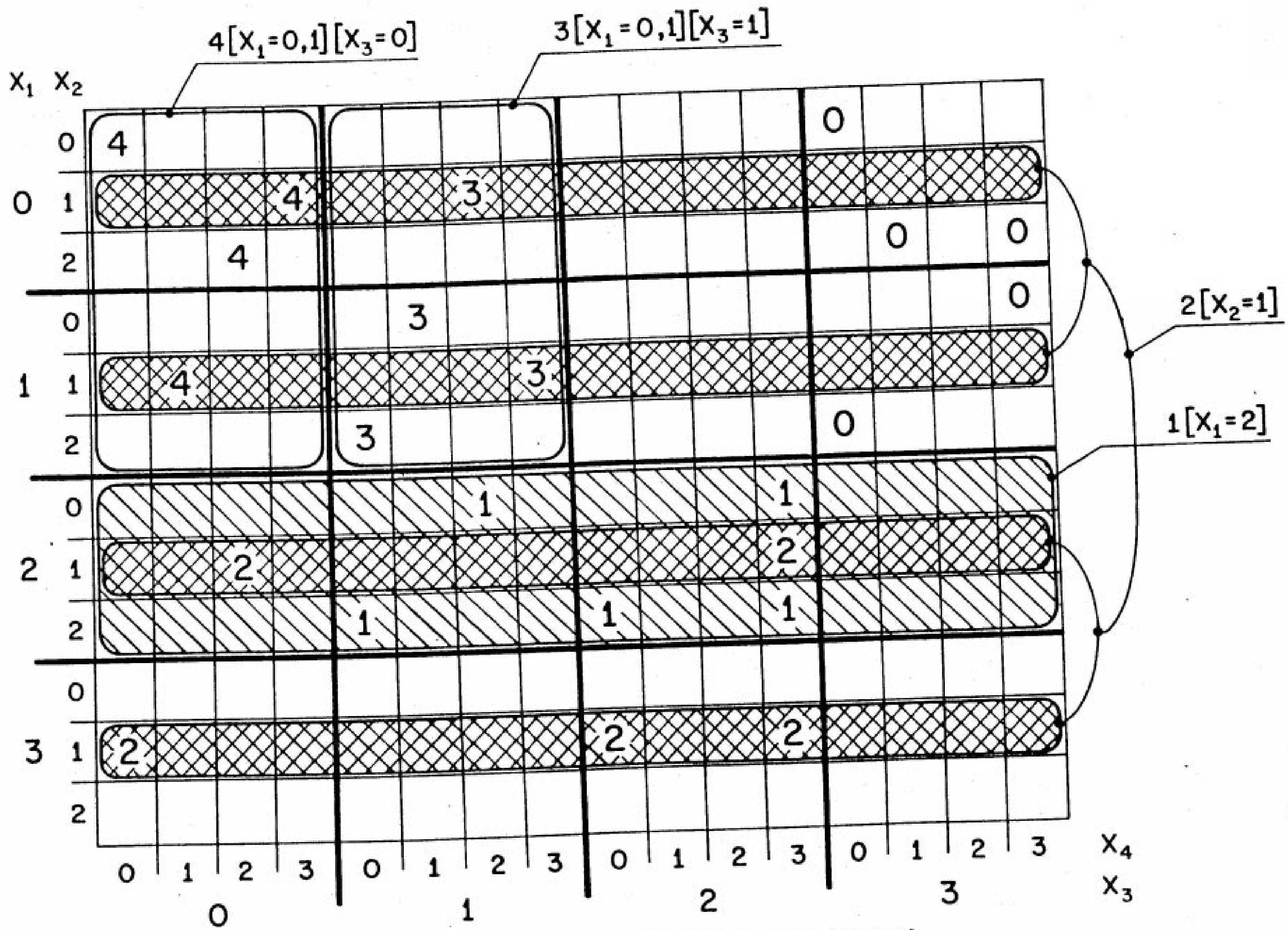
IC mode: Each complex does not cover any events from any event sets other than the one which it was specified to cover. The complexes which cover event sets F^2 and F^3 intersect with the complex which covers event set F^0 in an unspecified area of the event space. It is thus possible that an event in 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.

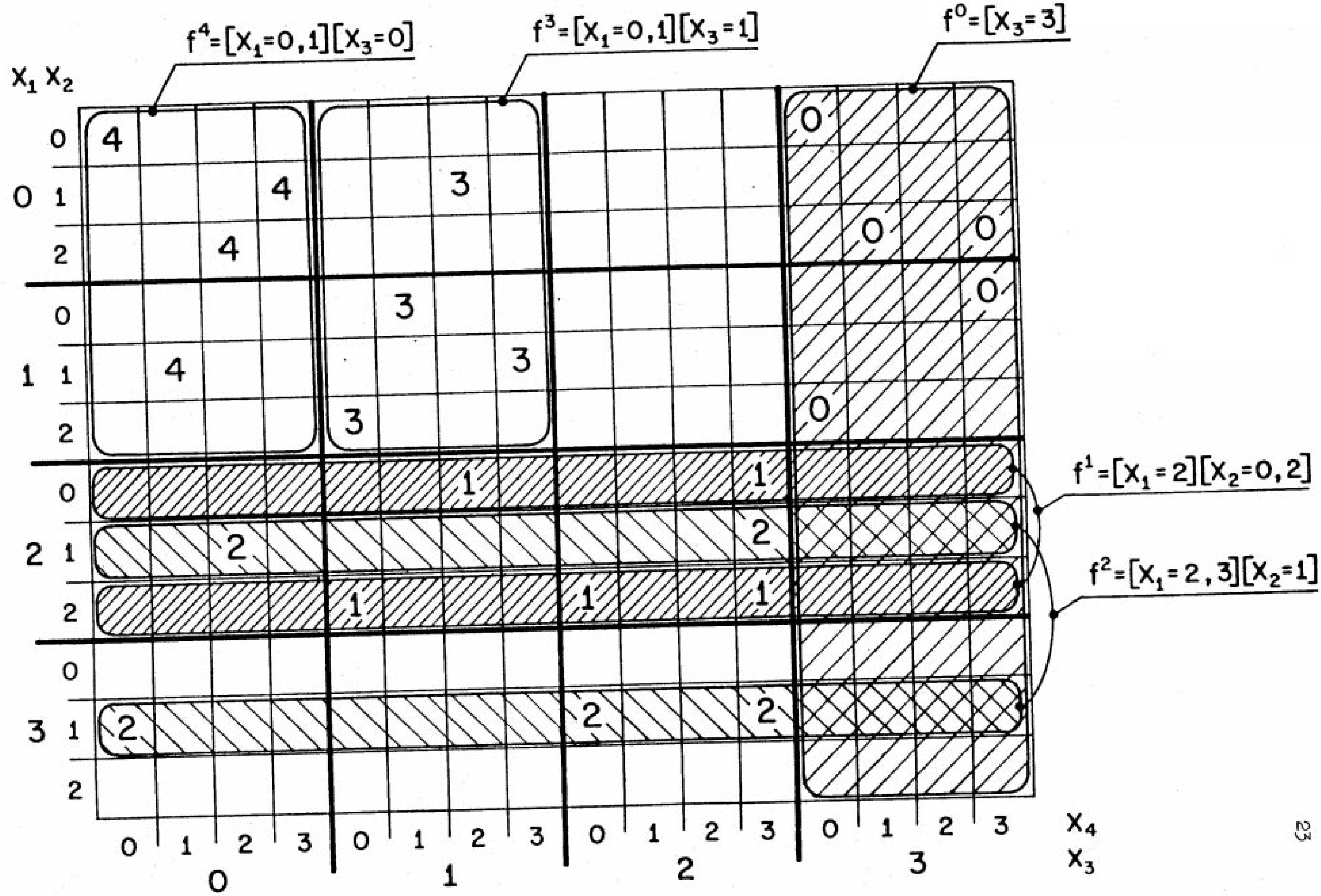
Estimates of Upper Bounds for Region and Time Requirements in Various Sizes of Problem (on IBM 360/75).

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

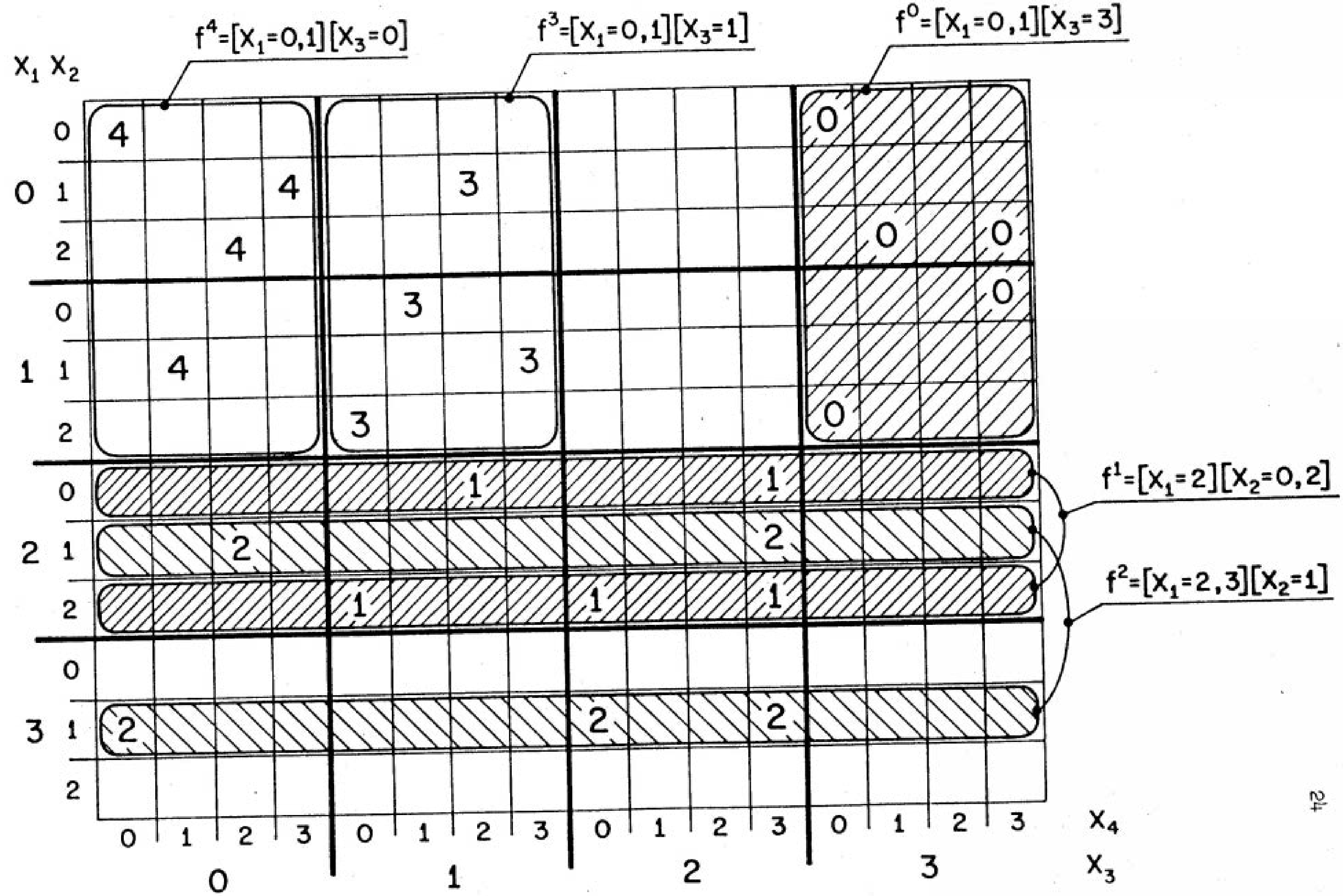
Figure 1



Graphical Representation of the VL_1 Formula Obtained



Graphical Representation of the Set of VL_1 Formulas
Obtained in Example 1 (IC mode).
Figure 3



Graphical Representation of the Set of VL_1 Formulas Obtained in Example 1 (DC mode).

```
// EXEC PCM=GEPAQ7,REGICN=116K
//STEPLIB  DC    DSN=LSER.F2123.GEPAC7,DISP=SHR
//SYSPRINT DD    SYSCLT=A
//COVER DC SYSOUT=B,CCB=(RECFM=FB,LRECL=80,BLKSIZE=80,BUFNO=1)
//SYSIN DD *
MODE='VL'
TITLE=3
NGF=100 MAXSTAR=100 CLTSTAR=90;
*************************************************************************
                    EXAMPLE # 1      MODE OF OPERATION IS'VL'
*************************************************************************
1 'FACTORS' 2

4
4, 3, 4 ,4

5         4 4 5 5 5

C 1 2 3 4

2         1 2          0 0

0 0 0 0,    0 2 0 2,    0 1 0 3,    1 1 0 1
0 1 1 2,    1 0 1 1,    1 1 1 ?,    1 2 1 0
2 1 0 2,    3 1 0 0,    2 1 2 3,    3 1 2 0,    3 1 2 3,
2 0 1 2,    2 2 1 0,    2 0 2 3,    2 2 2 0,    2 2 2 3
0 0 ? 0,    0 2 3 1,    0 2 3 3,    1 0 3 3,    1 2 3 0
```

Input Description

1. JCL
2. Control parameters (those not specified assume default values)
3. Title (3 lines of the title)
4. NSPEC 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

(11) Control parameters

```
MODE='IC'
TITLE=3
NGE=100 MAXSTAR=100 CUTSTAR=90;
```

(12) Data parameters

```
************************************************************************
                   EXAMPLE # 1  MODE = 'IC'
************************************************************************
1 'FACTORS' 2
4
4, 3, 4 ,4
5 4 4 5 5 5
0 1 2 3 4
2 1 2 0 0
                                        1 1 0 1
0 0 0 0,    0 2 0 2,    0 1 0 3,    1 2 1 0
0 1 1 2,    1 0 1 1,    1 1 1 3,    3 1 2 0,    3 1 2 3,
2 1 0 2,    3 1 0 0,    2 1 2 3,    2 2 2 0,    2 2 2 3
2 0 1 2,    2 2 1 0,    2 0 2 3,    1 0 3 3,    1 2 3 0
0 0 3 0,    0 2 3 1,    0 2 3 3,
```

(13) Control parameters

```
MODE='DC'
TITLE=3
NGE=100 MAXSTAR=100 CUTSTAR=90;
```

(14) Data parameters

```
************************************************************************
                   EXAMPLE # 1       MODE OF OPERATION IS 'DC'
************************************************************************
1 'FACTORS' 2
4
4, 3, 4 ,4
5 4 4 5 5 5
0 1 2 3 4
2 1 2 0 0
                                        1 1 0 1
0 0 0 0,    0 2 0 2,    0 1 0 3,    1 2 1 0
0 1 1 2,    1 0 1 1,    1 1 1 3,    3 1 2 0,    3 1 2 3,
2 1 0 2,    3 1 0 0,    2 1 2 3,    2 2 2 0,    2 2 2 3
2 0 1 2,    2 2 1 0,    2 0 2 3,    1 0 3 3,    1 2 3 0
0 0 3 0,    0 2 3 1,    0 2 3 3,
```

```
***************************************************************************
                    EXAMPLE # 1    MODE OF OPERATION IS'VL'
***************************************************************************

SPACE ALLOCATED FOR G(E) IS  100                SPACE ALLOCATED FOR MQ IS   25

MAXIMUM STAR SIZE BEFCRE TRIMMING WILL BE DCNE =   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   SAVELQ=  0

INPUT FORMAT IS VECTCR

THE FOLLOWING VARIABLES ARE COVERED BY FACTCRS          2
THE FOLLOWING VARIABLES ARE COVERED BY INTERVALS    1        3   4

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  **


OLIST=    0   1   2   3   4

NCRIT=    2
CLIST=    1       2
TLIST=    0.00    0.00

LQ-STAR OPTION  LOST=   '1'B

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

CLASS F( 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
```

Output for Example #1

```
    EVENT NO.   5=    3    1    2    3

:LASS 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           COV  NEW  IND  TOT

COMPLEX:      (X1= 0 1 )   (X3= 0 )                                           4    4    4    4


**** 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           COV  NEW  IND  TOT

COMPLEX:      (X1= 0 1 )   (X3= 1 )                                           4    4    4    4


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

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


THE FOLLOWING  1 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS  2           COV  NEW  IND  TOT

COMPLEX:      (X2= 1 )                                                        5    5    5    5


**** 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          COV  NEW  IND  TCT
COMPLEX:      (X1= 2 )                                                        5    5    5    5

**** 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 G(E) IS  100                    SPACE ALLOCATED FOR MQ IS   25

MAXIMUM STAR SIZE BEFORE TRIMMING WILL BE DONE =   100  THEN IT WILL BE CUT TO    90  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 VECTCR

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

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  **


OLIST=    0  1   2   3   4

NCRIT=    2
CLIST=    1       2
TLIST=    0.00    0.00

LO-STAR OPTION  LOST=    '1'B

CLASS F( 0)
      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 F( 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 F( 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
```

```
     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 FOLLCWING  1 CARTESIAN CCMPLEXES COVER THE EVENTS IN CLASS  0          COV  NEW  IND  TOT    to be covered

COMPLEX:     (X1= 0 1 )   (X3= 0 )                                          4    4    4    4

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

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


THE FOLLOWING  1 CARTESIAN CCMPLEXES COVER THE EVENTS IN CLASS  1          COV  NEW  IND  TOT

COMPLEX:     (X1= 0 1 )   (X3= 1 )                                          4    4    4    4

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

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


THE FOLLOWING  1 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS  2          COV  NEW  IND  TOT

COMPLEX:     (X1= 2 3 )   (X2= 1 )                                          5    5    5    5

**** 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                COV  NEW  IND  TOT
COMPLEX:     (X1= 2 )   (X2= 0 2 )                                                 5    5    5    5

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

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


THE FOLLOWING  1 CARTESIAN CCMPLEXES COVER THE EVENTS IN CLASS  4                COV  NEW  IND  TOT
COMPLEX:     (X3= 3 )                                                              5    5    5    5

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

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

*** NORMAL TERMINATION ***
```

```
*****************************************************************************
                    EXAMPLE # 1        MODE OF OPERATION IS 'DC'
*****************************************************************************

SPACE ALLOCATED FOR G(E) IS  100                  SPACE ALLOCATED FOR MQ 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   SAVELQ=  0

INPUT FORMAT IS VECTOR

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

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  **


OLIST=    0    1    2    3    4

NCRIT=    2
CLIST=    1       2
TLIST=    0.00    0.00

LQ-STAR OPTION  LQST=   '1'B

CLASS F( 0)
     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 F( 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 F( 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
```

```
    EVENT NO.   5=    3    1    2    3

LASS 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

LASS 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

'HE FOLLOWING  1 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS  0          COV  NEW  IND  TOT
                                                                            4    4    4    4
:OMPLEX:    (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          COV  NEW  IND  TOT
                                                                            4    4    4    4
:OMPLEX:    (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          COV  NEW  IND  TOT
                                                                            5    5    5    5
:OMPLEX:    (X1= 2 3 )   (X2= 1 )

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

THE LARGEST STAR HAC    4 ELEMENTS; THE LARGEST INTERMEDIATE STAR HAD    4 ELEMENTS
```

```
THE FOLLOWING  1 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS  3                 COV  NEW  IND  TOT
COMPLEX:      (X1= 2 )   (X2= 0 2 )                                               5    5    5    5

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

THE LARGEST STAR HAD    1 ELEMENTS: THE LARGEST INTERMEDIATE STAR HAD    2 ELEMENTS


THE FOLLOWING  1 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS  4                 COV  NEW  IND  TOT
COMPLEX:      (X1= 0 1 )   (X3= 3 )                                               5    5    5    5

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

THE LARGEST STAR HAD    1 ELEMENTS: THE LARGEST INTERMEDIATE STAR HAD    2 ELEMENTS

*** NORMAL TERMINATICN ***
```

```
(1)  // EXEC PGM=GEPAQ7,REGION=116K
     //STEPLIB  DD   DSN=USER.P2123.GEPAQ7,DISP=SHR
     //SYSPRINT DD   SYSOUT=A
     //COVER DD SYSOUT=B,DCB=(RECFM=FB,LRECL=80,BLKSIZE=80,BUFNO=1)
     //SYSIN DD *
(2)  NMQ=25 , TITLE=3;
(3)  *****************************************************************
                     EXAMPLE       #     2
     *****************************************************************

(4)  1 'INTERVAL' 3

(5)  4       5 3 5 4

(6)  3       2 4 1

(7)  1 2 0

(8)  7     1 2 3 5 4 7 6

(9)   0 0 .1 .1 .5 .3 .75

(10) 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 0

(11) Z(2)=5.0    Z(3)=2.1  ;

(12) W(1,4)=1.2   W(2,1)=2.53   W(1,1)=1.44  ;
```

(1) JCL

(2) Control parameters (those not specified assume default values)

(3) TITLE (3 lines of the title)

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

(5) NV 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

(6) 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

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

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

(9) 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)

(10) Event sets

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

(12) Parameters for cirterion #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

Input Data for Example #2

	Input		Description

(1)	`// EXEC PGM=GEPAQ7,REGION=116K`
	`//STEPLIB DD DSN=USER.P2123.GEPAQ7,DISP=SHR`
	`//SYSPRINT DD SYSOUT=A`
	`//COVER DD SYSOUT=B,DCB=(RECFM=FB,LRECL=80,BLKSIZE=80,BUFNO=1)`
	`//SYSIN DD *`
		(1)	JCL

(2)	`NMQ=25 , TITLE=3;`	(2)	Control parameters (those not specified assume default values)

(3)	`****************************************************************`
	`EXAMPLE # 2`
	`****************************************************************`
		(3)	TITLE (3 lines of the title)

(4)	`1 'INTERVAL' 3`	(4)	NSPEC TYPE TYPELIST: variable type specification (x3 is an interval variable, the rest are FACTOR)
(5)	`4 5 3 5 4`	(5)	NV 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
(6)	`3 2 4 1`	(6)	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)
(7)	`1 2 0`	(7)	OLIST: ordering of event sets: (the second set is covered first, next the first set, and finally the third)
(8)	`7 1 2 3 5 4 7 6`	(8)	NCRIT CLIST: cost functional specification (all 7 cost criteria are to be used in the order given)
(9)	`0 0 .1 .1 .5 .3 .75`	(9)	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)
(10)	`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 0`	(10)	Event sets
(11)	`Z(2)=5.0 Z(3)=2.1 ;`	(11)	Parameters for criterion #3: weights of variables (variables x2 and x3 have the assigned weights, the rest (i.e. x1) have weight 1)
(12)	`W(1,4)=1.2 W(2,1)=2.53 W(1,1)=1.44 ;`	(12)	Parameters for cirterion #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)

Input Data for Example #2
Figure 7

```
******************************************************************
                    EXAMPLE        #   2
******************************************************************

SPACE ALLOCATED FOR G(E) 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       4
THE FOLLOWING VARIABLES ARE COVERED BY INTERVALS           3

NUMBER OF VARIABLES =   4

NUMBER OF LEVELS FOR EACH VARIABLE:  5  3  5  4

NUMBER OF EVENTS SPECIFIED FOR EACH CLASS:
       CLASS    #EVENTS
         0         2
         1         4
         2         1

                        ** AMOUNT OF UNUSED CORE =    OK  **


OLIST=    1   2   0

NCRIT=    7
CLIST=    1       2       3       5       4       7       6
TLIST=    0.00    0.00    0.10    0.10    0.50    0.30    0.75

LQ-STAR OPTION  LOST=   '1'B

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

CLASS F( 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 F( 2)
      EVENT NO.   1=    0     0     0     0

Z(1)= 1.00000E+00       Z(2)= 5.00000E+00       Z(3)= 2.09999E+00       Z(4)= 1.00000E+00;

W(0,1)= 1.00000E+00     W(0,2)= 1.00000E+00     W(0,3)= 1.00000E+00     W(0,4)= 1.00000E+00     W(1,1)= 1.43999E+00
W(1,2)= 1.00000E+00     W(1,3)= 1.00000E+00     W(1,4)= 1.19999E+00     W(2,1)= 2.52999E+00     W(2,2)= 1.00000E+00
W(2,3)= 1.00000E+00     W(2,4)= 1.00000E+00;
```

```
HE FOLLOWING  2 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS  1                COV  NEW  IND  TOT
OMPLEX:     (X1= 1 2 )   (X4= 0 1 )                                              3    3    3    3

OMPLEX:     (X2= 2 )   (X4= 2 )                                                  1    1    1    4

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

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


'HE FOLLOWING  1 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS  2.              COV  NEW  IND  TCT
:OMPLEX:     (X1= 0 )                                                            1    1    1    1

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

'HE LARGEST STAR HAC   14 ELEMENTS; THE LARGEST INTERMEDIATE STAR HAD   27 ELEMENTS


THE FOLLOWING  2 CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS  0               COV  NEW  IND  TOT
:OMPLEX:     (X4= 3 )                                                            1    1    1    1

:OMPLEX:     (X2= 1 )   (X4= 2 )                                                 1    1    1    2

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

THE LARGEST STAR HAC   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 DVL_1 formulas is algorithm A^q 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 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 (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.

$$\sum_{i=1}^{NV} NLEV(i)$$

(NV -- 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 $(j+1)^{th}$ 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:

$$'\underbrace{100}_{x1}\;\underbrace{0001}_{x2}\;\underbrace{0010}_{x3}\;\underbrace{00001}_{x4}'B$$

Clearly, each substring in a string representing an event has <u>exactly</u> one bit set to 1, and each substring in a string representing a non-empty complex has <u>at least</u> 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

$$(x1=2)(x4=1,3)$$

is represented as:

$$'\underbrace{001}_{x1}\;\underbrace{1111}_{x2}\;\underbrace{1111}_{x3}\;\underbrace{01010}_{x4}'B$$

5.3 Internal structure for storing intermediate stars

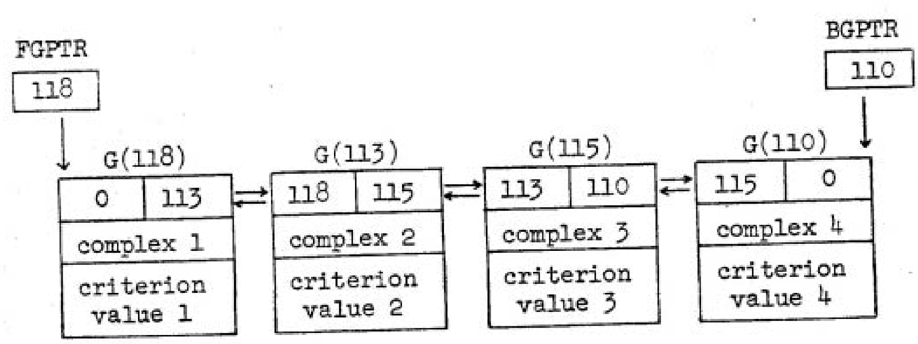
A star G(e|E) of an event e against event set E is generated in steps, each step corresponds to an event in E (more precisely, each step involves the multiplication of the current set of complexes, called an 'intermediate star', by complexes resulting from the extension of e against the consecutive event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

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 <u>head</u> of the list) and a pointer which points to the last element of the list (the <u>tail</u> 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 NFGPTR and NLGPTR),
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:



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 (TYPCOV) where '0'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 OLIST). 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 F1NOMQ and F1NOSTAR 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 F1NOMQ indicates elements which belong to the set E_1 and the array F1NOSTAR indicates the elements which belong to the set E_p in the algorithm A^q presented in Figure 1 of paper[2]). A one ('1'B) in a position of F1NOMQ indicates that the corresponding event of E belongs to E_1, i.e., it has not yet been covered by any quasi-optimal complex (L^q); a one ('1'B) in a position of F1NOSTAR 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 F1NOMQ and F1NOSTAR (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 FINOSTAR. A star is generated for this event and the 'best complex' (quasi-extremal LQ) 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 F1NOSTAR (i.e., events of E_p) has been exhausted, the process is repeated with the set represented by F1NOMQ (i.e., set E_1). 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 PRCOVER. 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 G(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 e 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 F1NOMQ) 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^q) (i.e., the number of events indicated by F1NOMQ which are covered by the complex).

BESTLQ -- This procedure uses the criteria specified in the input to select the 'best' complex (quasi-extremal L^q) 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^q.

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 auxilliary procedure used by STAR in the process of combining stars (application of absorption laws).

CKBE -- is an auxilliary procedure used by STAR in the process of combining stars.

REMOVEGN(REMOVEG) -- 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.

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

FRECORE -- 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/1 (AQ7) is currently the main and most developed program related to the computer implementation of VL_1. Other programs which are currently available include:

- OMNIBUS -- A program which can:

  A. Evaluate a given VL_1 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 VL_1 formulas used as classification rules*.

*This last function is implemented in an extended form in the program CONFUS (see next).

- UNICLASS (AQ8) -- A special program developed for the synthesis of VL_1 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.[21]

- AQVAL/1 (AQ9) -- A program which synthesizes quasi-optimal DVL_1 formulas from non-optimal DVL_1 formulas.[20]

- CONFUS -- A special program implementing various methods of evaluation of families of DVL_1 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)

- SIM1 -- A program which detects symmetry in (incompletely specified) VL_1 functions. (This program is still under development.)

REFERENCES

1. Michalski, R. S., "VARIABLE-VALUED LOGIC: System VL_1," 1974 International Symposium on Multiple-Valued Logic, West Virginia University, Morgantown, West Virginia, May 29-31, 1974.

2. Michalski, R. S., "Synthesis of Optimal and Quasi-Optimal Variable-Valued Logic Formulas," submitted for presentation at the 5th International Symposium on Multiple-Valued Logic, Bloomington, Indiana, May 13-16, 1975.

3. Michalski, R. S., "On the Quasi-Minimal Solution of the General Covering Problem, Proceedings of the V International Symposium on Information Processing (FCIP 69), Vol. A3 (Switching Circuits), Yugoslavia, Bled, October 8-11, 1969.

4. Michalski, R. S., "Automatic Synthesis of the Quasi-Minimal Multiple-Output Switching Circuits," Proceedings of the VI Yugoslav International Symposium on Information Processing (FCIP), Yugoslavia, Bled, September 23-26, 1970.

5. Michalski, R. S. and Z. Kulpa, "A System of Programs for the Synthesis of Combinatorial Switching Circuits Using the Method of Disjoint Stars," Proceedings of the IFIP Congress 71, Yugoslavia, August 23-28, 1971.

6. Michalski, R. S. and B. H. McCormick, "Interval Generalization of Switching Theory," Proceedings of the Third Annual Houston Conference on Computer and System Science, Houston, Texas, April 26-27, 1971.

7. Michalski, R. S. and B. H. McCormick, "Interval Generalization of Switching Theory," (an extended version of paper 6), Report No. 442, Department of Computer Science, University of Illinois, Urbana, Illinois, May 3, 1971.

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

9. Michalski, R. S., "A Method for Quasi-Minimal Solution of the General Covering Problem and its Application to the Synthesis of Automata," Avtomatica i Telemekhanica, No. 1, Academy of Sciences USSR, 1972 (in Russian).

10. Michalski, R. S., "A Variable-Valued Logic System as Applied to Picture Description and Recognition, GRAPHIC LANGUAGES, Proceedings of the IFIP Working Conference on Graphic Languages, Vancouver, Canada, May 1972 (in English).

11. Michalski, R. S., "Discovering Classification Rules by the Variable-Valued Logic System VL_1," Proceedings of the Third International Joint Conference on Artificial Intelligence, Stanford, California, August 20-24, 1973.

12. Michalski, R. S., "AQVAL/1--Computer Implementation of a Variable-Valued Logic System and the Application to Pattern Recognition," Proceedings of the First International Joint Conference on Pattern Recognition, Washington, D.C., October 30-November 1, 1973.

13. Michalski, R. S., "A Variable-Decision Space Approach for Implementing a Classification System," 2nd International Joint Conference on Pattern Recognition, August 13-15, 1974, Lyngby-Copenhagen, Denmark.

14. Michalski, R. S., "Learning by Inductive Inference," NATO Advanced Study Institute on Computer Oriented Learning Processes, Aug. 26-Sept. 7, 1974, France.

15. Michalski, R. S., "VARIABLE-VALUED LOGIC: Recent Developments," Twelfth Annual Allerton Conference on Circuit and System Theory, Allerton House, Monticello, Illinois, October 2-4, 1974.

16. Lovby, Terje, "Variable-Valued Logic Applied to Quality Control in a Clinical Laboratory," Master's Thesis, Department of Computer Science, University of Illinois, Urbana, Illinois, 1974.

17. Michalski, R. S., "Problems of Designing an Inferential Medical Consulting System," First Illinois Conference on Medical Information Systems, Urbana, Illinois, October 17-18, 1974.

18. Baskin, A. B., "A Comparative Discussion of Variable-Valued Logic and Grammatical Inference," Report No. 663, Department of Computer Science, University of Illinois, Urbana, Illinois, July 1974.

19. Jensen, Gerald M., "Preliminary Report of Program That Detects Symmetry of Variable-Valued Logic Functions," Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois.

20. Cuneo, Roland Philippe, "Selected Problems of Minimization of Variable-Valued Logic Formulas," Master's Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois.

21. Yuen, H., "Uniclass Cover Synthesis: User's Guide," Internal report.

APPENDIX

Steps in input preparation
(short form)

		Page
I. Specify control parameters:		3
• MODE	('IC')*	3
• INFORM	('VECTOR')	6
• TITLE	(3)	6
• MAXSTAR	(150)	6
• CUTSTAR	(50)	6
• NGE	(200)	7
• NRQ	(25)	7
• LQST	('1'B)	7
• SAVE	('0'B)	8
• QLQT	('0'B)	8
• LQTRACE	('0'B)	8
• STRACE	('0'B)	8
• QST	('0'B)	9
II. Type semi-colon (;)		9
III. Type the title of the problem		9
IV. Specify data parameters:		9
• NSPEC TYPE TYPELIST(NSPEC) (type of variables, e.g., 2 'FACTORS' 1 4)		9
• NV NLEV(NV) (# of variables and size of their domains, e.g., 4 5, 3, 2, 4)		10
• NCL NE(NCL) (# of event sets and their size, e.g., 3 5, 1, 2)		10
• OLIST (desired order of event sets, e.g., 1 0 2)		11
• NCRIT CLIST(NCRIT) TLIST(NCRIT) (criteria list and tolerance list, e.g., 3 1, 2, 4 0, 0, 0.1)		11
• F (list of event sets, e.g., 0 0 1 0 2 0 1 3 2 1 0 3 1 1 3 2 0 2)		14
• Z (list of costs for variables, e.g., Z(3) = 5;)		14
• W (list of weights for events, e.g., W(1,3) = 5 W(0,1) = 2;)		15

*Numbers in parenthesis are default values for parameters.

List of User Optimization Criteria

Criterion #	Brief Description
1	Number of terms in a formula (t)
2	Number of selectors in a formula (s)
3	Cost of variables in a formula (z)
4	Degree of generalization (g)
5	Weight of events covered by a formula (w)
6	Length of references (l)
7	Relative scope of references (r)

APPENDIX B

PROGRAM LISTING

SOURCE LISTING

```
STMT LEV NT

 1     0  |      AQ7: PROC OPTIONS(MAIN); /** AQ-VERSION AQ-7 **/
          |
          |/*  THIS PROCEDURE DETERMINES A SET OF COMPLEXES WHICH WILL
          |    COVER FLE 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, FGPTR AND LGPTR, POINT TO THE BEGINNING AND
          |     ENDING ELEMENTS OF THE CURRENTLY COMPLETE INTERMEDIATE STAR.
          |     THE THIRD PAIR, NFGPTR AND NLGPTR, 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:
          |                '1000000000010010000000010'B.
          |     INTERVALS ARE PAIRS OF VECTORS AND ARE STORED TO SHOW THE
          |     UPPER AND LOWER BOUND OF EACH VARIABLE.  THUS 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
          |                '0111001111110010000000110'B.
          |
          |     ALL INPUT IS FREE FORMAT IN THE FOLLOWING ORDER:
          |     (NGE, NMQ, MAXSTAR, CUTSTAR, LQTRACE, STRACE, QST, QLQT, INFORM,
          |     TITLE) ALL OPTIONAL & IN PL1 DATA FORMAT, A SEMICOLON (MUST APPEAR
          |     DUE TO THE PREVIOUS VARIABLES BEING READ IN IN DATA FORM),
```

```
STMT LEV NT

          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, FIE, FOE.
          NGE, NMQ, MAXSTAR, CUTSTAR, & TITLE ARE INTEGER VALUES IN DATA
               FORMAT,
          NV, NL, NI, & NO ARE INTEGER VALUES IN LIST FORMAT,
          LQTRACE, STRACE, QST, QLQT, & 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,
          FIE AND FOE 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      QLQT='1'B,           INFORM = 'VECTOR'    ;
            2  'FACTORS'   2,3,
           3,  4          2   3,
               0,1,2,      3  2   3      1    0    3  3 2 0    0  3, 3
             NMQ=20,     STRACE = '0'B    INFORM='GAMMA' ;
               0 'INTERVALS'
               3 4,  2   3
                   6    59          19  56,   15

               */

 2   1  0      DECLARE
               (LQTRACE,STRACE,QST,QLQT,FMIX,SAVE) BIT(1) ALIGNED STATIC,
                  SAVELQ BIT(1) ALIGNED STATIC,
                     MODE CHAR(2) STATIC,
                     (INFORM CHAR(6), LINE CHAR(132) VARYING) STATIC,
               (NV,NLMAX,LEN,NGE,NMQ) FIXED BINARY(15) STATIC,
               (I,IO,IG,IKL,J,NUM) FIXED BINARY(15) STATIC,
               (GPTR,FGPTR,LGPTR,NFGPTR,NLGPTR) FIXED BINARY(15) STATIC,
               (NEXTFG,LASTFG) FIXED BINARY(15) STATIC,
               (MQPTR,DELTA,MQPTS) FIXED BINARY(15) STATIC,
               (IIL,LQ,NFILQ,NOFILQ) FIXED BINARY(15) STATIC,
               (NES,LNES,LNEAL,LSS,CVNUSTAR) FIXED BINARY(15) STATIC,
               (MAXSTAR,CUTSTAR,NUMTRIM,NUMSTRIM) FIXED BINARY(15) STATIC,
                TITLE FIXED BINARY(15) STATIC,
                CHARBUF CHARACTER(80) STATIC,
               COVER FILE STREAM OUTPUT EXTERNAL ;
 3   1  0      DCL( TYPCOV(64) BIT(1), LQST BIT(1),
                   NSPEC FIXED BINARY(15),
                  TYPE CHAR(9) )STATIC,
```

```
        (K,MAXCL,NCL,NEMAX,NECLMAX,NE(0:50),NDOWN)
           FIXED BINARY(15) ;

/*  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'B
  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-0,...,F-MAXCL
  FGPTR        - (FIRST G POINTER) POINTS TO THE FIRST COMPLEX OF
                 OF THE CURRENT "STAR"
  FMIX         - (FIRST MIX CALL)  USED TO DIFFERENTIATE BETWEEN
                 FIRST (FMIX='1'B) OR LATER (FMIX='0'B) 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 = X(NV) ;
                       DO I = 1 TO NV-1 ;
                             PROD = 1 ;
                             DO K = 0 TO K-1 ;
                                   PROD = PROD * NLEV(NV-K) ;
                             END ;
                             J = J + X(NV-I) * PROD ;
                       END ;
                 OR WHEN NLEV(1)=NLEV(2)= ... = NLEV(NV-1), THEN
                       J = 0 ;
                       DO I = 0 TO NV-1 ;
                             J = J + X(NV-I) * NL**I ;
                       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 FOE ARRAY
  I1           - USED AN AN INDEX OVER THE F1E ARRAY
```

```
I1L        - INDEX USED TO POINT TO THE ELEMENT OF F1 LAMBDA
             FOR WHICH A STAR IS BEING GENERATED
J          - USED AS AN INDEX OVER THE NEW COMPLEXES THAT ARE TO
             BE MULTIPLIED TIMES THE CURRENT STAR
K          - CURRENT D-LIST ELEMENT: K=DLIST(DLPTR)
LASTFG     - (LAST FREE G) POINTS TO THE LAST G(E) ELEMENT IN
             THE "FREE" LIST
LEN        - (LENGTH) LENGTH OF THE UNARY BIT STRING USED TO
             REPRESENT ELEMENTS
LGPTR      - (LAST G POINTER) POINTS TO THE LAST COMPLEX OF
             THE CURRENT "STAR"
LNEAL      - LARGEST NUMBER OF ELEMENTS IN ANY INTERMEDIATE STAR
LNES       - LARGEST NUMBER OF ELEMENTS IN EACH INTERMEDIATE STAR
LQ         - POINTER TO AN ENTRY OF G(E) CONSIDERED "BEST" OF
            THE STAR ACCORDING TO GIVEN CRITERIA
LQTRACE    - VARIABLE WHICH ALLOWS A PRINTED TRACE OF QUASI-
              EXTREMAL LQ RESULTS (LQTRACE='1'B).  THE ORIGINAL
              STAR, THE QUASI-EXTREMAL, THE NUMBER OF UNCOVERED
              EVENTS OF F1 LAMBDA COVERED BY LQ, AND THE STATE
              OF THE FINOSTAR BIT ARRAY ARE PRINTED.
              THIS TRACE IS SUPPRESSED IF LQTRACE='0'B.
LSS        - LARGEST STAR SIZE IN NUMBER OF ELEMENTS
MAXSTAR    - LARGEST NUMBER OF ELEMENTS THAT A STAR IS ALLOWED
              TO HAVE BEFORE TRIMMING IS APPLIED
MAXCL        - HIGHEST CLASS NUMBER
   MODE -        VL OR IC MODE
NECLMAX     - NCL*NEMAX, I.E. HIGHEST DIMENSION OF ARRAY FDOWN
NCL        - # OF CLASSES (=MAXCL+1)
MQPTR      - (MQ POINTER)  POINTS TO THE LAST COMPLEX IN MQ
NES        - NUMBER OF ELEMENTS IN A STAR
NEXTFG     - (NEXT FREE G)  POINTS TO THE NEXT G(E) ELEMENT IN
              THE "FREE" OR UNUSED LIST OF G(E)
NE         - (NUMBER OF ELEMENTS IN F) NE(I)=# OF ELEMENTS
              IN CLASS F-I, FOR I=0,...,MAXCL
NEMAX      - LARGEST COMPONENT OF NE
NFGPTR     - (NEXT FIRST G POINTER)  POINTS TO THE FIRST
              COMPLEX OF THE NEW STAR
CVNOSTAR       - NUMBER OF EVENTS NEVER COVERED BY ANY STAR
               (I.E. NUMBER OF EVENTS IN FINOSTAR)
NF1LQ      - NUMBER OF F1 ELEMENTS COVERED BY PRESENT LQ
NGE        - (NUMBER OF G(E) ELEMENTS)  TOTAL NUMBER OF G(E)
              IN THE FREE LIST PLUS THE USED LIST
NLEV          - NUMBER OF LEVELS FOR EACH VARIABLE
NLGPTR     - (NEXT LAST G POINTER)  POINTS TO THE LAST COMPLEX
              OF THE NEW STAR
NMQ        - (NUMBER OF MQ)  MAXIMUM NUMBER OF COMPLEXES WHICH
              CAN BE STORED IN LIST MQ
REMUNCOV       - NEW NUMBER OF ELEMENTS IN FINOSTAR THAT ARE UNCOVERED
NOF1LQ     - NUMBER OF ELEMENTS IN THE ORIGINAL F1 COVERED
```

```
STMT LEV NT

              |                     BY PRESENT LQ
              |   NUM             - NUMBER OF COMPLEXES RESULTING FROM A CALL TO MIX
              |   NSPEC           - NO. OF VARIABLES SPECIFIED TO BE COVERED BY
              |                     FACTORS RESP. INTERVALS (INPUT DATA)
              |   NUMSTRIM        - 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 LQ TRACE) VARIABLE WHICH ALLOWS A MINIMAL
              |                     TRACE OF THE LQ DETERMINATION PROCESS.  SIZE OF THE
              |                     STAR, NUMBER OF ELEMENT OF F1 COVERED BY LQ, AND
              |                     COUNT OF REMAINING ELEMENTS OF F1NOSTAR ARE PRINTED AFTE|R
              |                     EACH LQ 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 MULTIPICATION 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 RETURNG 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'
              |   TYPCOV          - (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; E.G.,
              |                     F1E(5) = (3,2,1,4) WOULD BE LISTED ON THE INPUT LINE AS
              |                             3    2    1    4
              |   ZEROS           - BIT STRING OF ALL 0 BITS
              |          *** */
              |
   4   1   0  |ON NAME(SYSIN) SNAP BEGIN;
   5   2   0  | DCL DATAFIELD BUILTIN;
   6   2   0  |PUT SKIP(2) EDIT('THE INVALID STRING ENCOUNTERED DURING THE "GET'
              |,' DATA" WAS "',DATAFIELD,'".','IT IS IGNORED:  FIX THIS AND ',
              |'PLEASE TRY THE RUN AGAIN.')
```

```
STMT LEV NT

             |(4 A, SKIP,COL(2), 2 A);
  7   2  0   |END;
             |
  8   1  0   |  ON ENDFILE(SYSIN) GO TO FINISH;
             |
  9   1  0   |NEWDATA:
 10   1  0   |          CUTSTAR=50;    MAXSTAR=150;
 11   1  0   |        NGE = 200 ;
 12   1  0   |        NMQ =  25 ;
 13   1  0   |          MODE='IC';
 14   1  0   |        LQTRACE,STRACE,QST,QLQT,SAVE,SAVELQ = '0'B ;
 15   1  0   |        INFORM = 'VECTOR' ;
 16   1  0   |        TITLE = 3 ; LQST='1'B;
             |
 18   1  0   |        GET DATA(NGE,NMQ,LQTRACE,STRACE,QLQT,QST,MODE,
             |             SAVE,SAVELQ,MAXSTAR,CUTSTAR,INFORM,TITLE,LQST);
 19   1  0   |        PUT PAGE;
 20   1  0   |        IF TITLE>0 THEN DO;
 21   1  1   |               PUT SKIP(2);
 22   1  1   |               GET SKIP(1);
 23   1  1   |               DO I=1 TO TITLE;
 24   1  2   |                      GET EDIT(CHARBUF)(A(80));
 25   1  2   |                      PUT SKIP(1) EDIT (CHARBUF)(COLUMN(20),A);
 26   1  2   |               END;
 27   1  1   |        END;
 28   1  0   |        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)) ;
 29   1  0   |        PUT SKIP(2) EDIT
             |             ('MAXIMUM STAR SIZE BEFORE TRIMMING WILL BE DONE =',
             |         MAXSTAR,' THEN IT WILL BE CUT TO',CUTSTAR,'  MODE= ',MODE)
             |             (A,F(6),A,F(6),A,A);
 30   1  0   |         IF INDEX('ICVLOC',MODE)=0 THEN
 31   1  1   |         DO; PUT SKIP LIST('      IC MODE ASSUMED'); MODE='IC';END;
 34   1  0   |        PUT SKIP(2) EDIT
             |             ('LQ TRACE =',LQTRACE,'STAR TRACE =',STRACE,
             |              'QUICK LQ TRACE =',QLQT,'QUICK STAR TRACE =',QST,
             |       'SAVE COVER DATA =',SAVE,'SAVELQ= ',SAVELQ)((6)(A,F(2),X(3)));
 35   1  0   |        PUT SKIP(2) EDIT('INPUT FORMAT IS ',INFORM)(A,A);
             |
             |              /* INTERVALS OR FACTORS */
             |
 36   1  0   |        GET LIST(NSPEC,TYPE);
 37   1  0   |        TYPCOV='0'B;
 38   1  0   |        IF NSPEC > 0 THEN DO I=1 TO NSPEC;
 39   1  1   |              GET LIST(J);
 40   1  1   |              TYPCOV(J)='1'B;
 41   1  1   |        END;
 42   1  0   |        IF SUBSTR(TYPE,1,3)='INT' THEN TYPCOV = ¬TYPCOV;
```

```
STMT LEV NT

 43   1  0 |        ELSE IF SUBSTR(TYPE,1,3) ¬= 'FAC' THEN PUT SKIP(2) LIST
           |        ('*** INVALID COVER TYPE SPECIFIED - FACTOR ASSUMED ***');
           |
 44   1  0 |        GET LIST(NV);
           |
 45   1  0 |        IF NSPEC=0 | NSPEC=NV THEN DO;
 45   1  1 |               LINE='ALL VARIABLES WILL BE COVERED BY ';
 47   1  1 |               IF TYPCOV(1) THEN LINE=LINE||'FACTORS';
 48   1  1 |               ELSE LINE=LINE||'INTERVALS';
 49   1  1 |               PUT SKIP(2) LIST(LINE);
 50   1  1 |        END;
 51   1  0 |        ELSE DO;
 52   1  1 |               PUT SKIP(2) LIST('THE FOLLOWING VARIABLES ARE '
           |                      ||'COVERED BY FACTORS');
 53   1  1 |DO I=1 TO NV; IF (50+(I-1)*4)>120 THEN PUT SKIP DATA;
 55   1  2 |               IF TYPCOV(I) THEN PUT EDIT (I)
           |                                      (COLUMN(50+(I-1)*4),F(3));
           |
 56   1  2 |END;
 57   1  1 |PUT SKIP LIST('THE FOLLOWING VARIABLES ARE '
           |                      ||'COVERED BY INTERVALS');
 58   1  1 |DO I=1 TO NV; IF (50+(I-1)*4)>120 THEN PUT
           |                                          SKIP(1) DATA;
 60   1  2 |               IF ¬TYPCOV(I) THEN PUT EDIT(I)
           |                               (COLUMN(50+(I-1)*4),F(3));
           |                                                    END;
 61   1  2 |END;
           |
 63   1  0 |        PUT SKIP(2) EDIT('NUMBER OF VARIABLES =',NV)(A,F(4));
           |
 64   1  0 |        IF CUTSTAR > MAXSTAR THEN
           |             DO ;
 65   1  1 |                  PUT SKIP(3) LIST
           |                      ('* CUTSTAR CANNOT BE GREATER THAN MAXSTAR *') ;
 66   1  1 |                  GO TO FINISH ;
 67   1  1 |             END ;
           |
 68   1  0 |        IF SAVE THEN OPEN FILE(COVER) LINESIZE(80) ;
           |
           |
 69   1  0 | MAINBLK: BEGIN REORDER;
           |
 70   2  0 |        DECLARE (NLEV(NV),NVP(NV)) FIXED BINARY(15);
           |
 71   2  0 |        GET LIST(NL);
 72   2  0 |        PUT SKIP(2) EDIT('NUMBER OF LEVELS FOR EACH VARIABLE:',NL)
           |               (A,(NV)F(3));
 73   2  0 |        LEN,NLMAX=0;
 74   2  0 |        DO I=1 TO NV;
 75   2  1 |               NVP(I)=LEN+1;
 76   2  1 |               LEN=LEN+NLEV(I);
```

```
STMT LEV NT

 77   2  1  |            IF NLMAX < NLEV(I) THEN NLMAX=NLEV(I);
 78   2  1  |         END;
            |
 79   2  0  |         GET LIST(NCL,(NE(I)DO I=0 TO NCL-1));
 80   2  0  |            MAXCL=NCL-1;
 81   2  0  |         PUT SKIP(2) EDIT('NUMBER OF EVENTS',
            |              ' SPECIFIED FOR EACH CLASS:','CLASS    #EVENTS',
            |              ( I,NE(I) DO I=0 TO MAXCL))
            |              (A,A,SKIP(1),COLUMN( 8),A,SKIP(1),
            |               (100)((2)F(10),SKIP(1)));
 82   2  0  |         NEMAX,NECLMAX=0;
 83   2  0  |         DO I=0 TO MAXCL;
 84   2  1  |            IF NEMAX < NE(I) THEN NEMAX=NE(I);
 85   2  1  |            NECLMAX=NECLMAX+NE(I);
 86   2  1  |         END;
            |
 87   2  0  |  BEGIN REORDER;
            |
 88   3  0  |         DECLARE  (E(0:MAXCL,1:NEMAX),
            |              FDOWN(1:NECLMAX)) BIT(LEN) ALIGNED,
            |            TAINT BIT(LEN) ALIGNED,
            |            (OLIST(1:NCL),CLIST(1:7),NCRIT)FIXED BINARY(15),
            |            (OLPTR,OMQPTR) FIXED BINARY(15),
            |            TLIST(1:7) DEC FIXED(5,2),
            |             REMUNCOV FIXED DECIMAL(3) STATIC,
            |             ZEROS BIT(LEN) INIT('0'B),
            |             ONESINIT(LEN) BIT(1) INIT((LEN)(1)'1'B),
            |             ONES BIT(LEN) DEFINED ONESINIT,
            |
            |            (INDEP(30),TOTCOV(30),NEWCOV(30),NUMCOV(30))
            |            FIXED BIN(15),(COMCOVD,ISPL) FIXED BIN(15),
            |         1  G(NGE),
            |            2  INT BIT(LEN) ALIGNED,       /*  INTERVAL   */
            |            2  VAL DEC FLOAT,              /*  CRITERION VALUE */
            |            2  FPTR FIXED BINARY(15), /*  FORWARD POINTER  */
            |            2  BPTR FIXED BINARY(15), /*  BACKWARD POINTER */
            |
            |         1  MQ(NMQ),
            |            2  INT BIT(LEN) ALIGNED,       /*  INTERVAL   */
            |            2  TRIM CHARACTER(1),          /*  TRIMMING INFO */
            |
            |         (INTP(NV),TINT,TAND) BIT(NLMAX) ALIGNED,
            |         (M(NV),FINOMQAD(NEMAX),FINOMQ(NEMAX),FINOSTAR(NEMAX)) BIT(1),
            |         (POSP(NV),KL(NV,NGE/2),KE(NV,NGE/2),KB(NV,NGE/2),
            |          KLC(NV),KEC(NV),KBC(NV)) FIXED BINARY(15) ,
            |         (W(0:MAXCL,1:NEMAX),Z(1:NV)) DEC FLOAT;
            |
            |/*  DEFINITION OF VARIABLES DECLARED IN THIS BLOCK:
```

STMT LEV NT

```
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 LQ FROM THE CURRENT STAR IN PROC BESTLQ.
FINOMQ        - BIT-ARRAY CONTAINING ELEMENTS OF E(K,*) WHICH
            HAVE NOT BEEN COVERED YET BY A QUASI-EXTREMAL.
            I.E. FINOMQ(*)='1'B <=> E(K,*) NOT YET COVERED.
FINOSTAR      - BIT ARRAY DENOTING ELEMENTS OF E(K,*) WHICH
            HAVE NEVER BEEN COVERED BY ANY STAR; I.E.,
            FINOSTAR(I1)='1'B MEANS THAT E(K,I1) HAS NEVER BEEN
            COVERED BY ANY STAR
FPTR      - (FORWARD POINTER) POINTS TO THE ELEMENT OF G
            FOLLOWING IT IN THE CURRENT LIST
FDOWN         - ARRAY OF ALL ELEMENTS AGAINST WHICH WE ARE COVERING
G         - LIST USED FOR STORING INTERVALS OF STARS
INT       - BI-UNARY BIT STRING USED TO REPRESENT AN INTERVAL
INTP      - BI-UNARY BIT STRING USED TO REPRESENT A LITERAL
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(I) IS THE COUNT OF THE NUMBER OF POINTERS IN KB(I,*)
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
KEC       - KEC(I) IS THE COUNT OF THE NUMBER OF POINTERS IN KE(I,*)
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(I) IS THE COUNT OF THE NUMBER OF POINTERS IN KL(I,*)
M         - 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.
NDOWN     - # OF ELEMENTS IN FDOWN
OLIST     - (ORDER LIST) LIST SPECIFYING THE ORDER IN WHICH
            THE CLASSES F-0, ..., F-MAXCL ARE CONSIDERED FOR
            COVERING.
OLPTR     - (O-LIST POINTER)
OMQPTR    - (OLD MQ-POINTER)
POSP      - 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
```

```
STMT LEV NT

                  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,'K  **')
                             (COLUMN(35),A,F(4),A) ;
 91   3  0              PUT SKIP(2) ;

                        /*  INITIALIZE VARIOUS VARIABLES AND CONSTANTS     */

 92   3  0              NEXTFG = 1 ;
 93   3  0              LASTFG = NGE ;
 94   3  0              DO IG = 1 TO NGE ;
 95   3  1                   G.FPTR(IG) = IG + 1 ;
 96   3  1                   G.BPTR(IG) = IG - 1 ;
 97   3  1              END ;
 98   3  0              FPTR(NGE) = 0 ;
 99   3  0              GET LIST((OLIST(I) DO I=1 TO NCL));
100   3  0              GET LIST(NCRIT,(CLIST(I)DO I=1 TO NCRIT));
101   3  0              PUT SKIP(2) EDIT
                              ('OLIST= ',OLIST,'NCRIT=',NCRIT,'CLIST=',
                               (CLIST(I)  DO I=1  TO NCRIT))
                              (A,(NCL)(X(2),F(2)),SKIP(2),A,F(5),SKIP(1),
                               A,(NCRIT)(F(5),X(3)) );
102   3  0              GET LIST((TLIST(I) DO I=1 TO NCRIT));
103   3  0              PUT SKIP EDIT('TLIST=',
                                 (TLIST(I) DO I=1 TO NCRIT)) (A,(NCRIT)F(8,2));
104   3  0                   PUT SKIP(2) LIST('LQ-STAR OPTION'
                                             ||'  LQST=',LQST);

                        /*  GET INPUT EVENTS         */
105   3  0              IF INFORM = 'VECTOR' THEN CALL READVEC;
106   3  0                  ELSE IF INFORM = 'GAMMA' THEN CALL READGAM ;
107   3  0                       ELSE DO ;
108   3  1                            PUT SKIP(3) LIST
                                          ('** INVALID EVENT FORMAT SPECIFIED  **') ;
109   3  1                            GO TO FINISH ;
110   3  1                       END ;

111   3  0              OLPTR=1; MQPTR=0;
113   3  0              NDOWN=0;
114   3  0                IF MODE='VL' THEN
                        DO I=NCL TO 1 BY -1;
115   3  1                    II=OLIST(I);
```

```
STMT LEV NT

 116   3  1  |              DO J=1 TO NE(II);
 117   3  2  |                   NDOWN=NDOWN+1;
 118   3  2  |                   FDOWN(NDOWN)=E(II,J);
 119   3  2  |              END;
 120   3  1  |         END;
             |
 121   3  0  |         Z=1.;
 122   3  0  |         W=1.;
 123   3  0  |         DO J=1 TO NCRIT;
 124   3  1  |              IF CLIST(J)=3 THEN DO;
 125   3  2  |                   GET DATA(Z);
 126   3  2  |                   PUT SKIP(2) DATA(Z);
 127   3  2  |              END;
 128   3  1  |         END;
             |
 129   3  0  |         DO J=1 TO NCRIT;
 130   3  1  |              IF CLIST(J)=5 THEN DO;
 131   3  2  |                   GET DATA(W);
 132   3  2  |                   PUT SKIP(2) DATA(W);
 133   3  2  |              END;
 134   3  1  |         END;
 135   3  0  |              OMQPTR=0;
 136   3  0  |NEXTOEL:  DO WHILE (OLPTR <=NCL);
 137   3  1  |         K=OLIST(OLPTR); ISPL=1;
 139   3  1  |          IF MODE='VL'&OLPTR=NCL THEN GO TO NTNTIC;
 140   3  1  |          IF MODE='IC' THEN DO;
 141   3  2  |              NDOWN=0;
 142   3  2  |              DO I=NCL TO 1 BY -1;
 143   3  3  |                   IF I¬=OLPTR THEN
 144   3  4  |                   DO J=1 TO NE(OLIST(I)); NDOWN=NDOWN+1;
 145   3  4  |                   FDOWN(NDOWN)=E(OLIST(I),J); END;
 147   3  3  |              END;
 148   3  2  |                             END;
 149   3  1  |          IF MODE='VL' THEN NDOWN=NDOWN-NE(K);
 150   3  1  |          IF  MODE='DC' THEN DO;
 151   3  2  |              NDOWN=OMQPTR;
 152   3  2  |              DO J=OMQPTR+1 TO MQPTR;
 153   3  3  |                   NDOWN=NDOWN+1;
 154   3  3  |                   FDOWN(NDOWN)=MQ.INT(NDOWN);
 155   3  3  |                   END;
 156   3  2  |              DO I=OLPTR+1 TO NCL;
 157   3  3  |                   DO J=1 TO NE(OLIST(I));
 158   3  4  |                        NDOWN=NDOWN+1;
 159   3  4  |                        FDOWN(NDOWN)=E(OLIST(I),J);
 160   3  4  |                        END;
 161   3  3  |                   END;
 162   3  2  |                             END;
 163   3  1  |          OMQPTR=MQPTR;
 164   3  1  |             DELTA=0;
```

```
STMT LEV NT

165    3  1  |           FINOSTAR,FINOMQ='1'B;
166    3  1  |              CVNOSTAR,NOFILQ,NUMSTRIM,LNEAL,LSS=0;
             | /* FIND NEXT ELEMENT OF FINOSTAR WHICH HAS NOT YET BEEN COVERED */
             |
167    3  1  |      DO IIL=1 TO NE(K);
168    3  2  |         IF FINOSTAR(IIL) THEN DO;
169    3  3  |            CALL STAR ;
170    3  3  |            CALL BESTLQ ;
171    3  3  |            IF QLQT THEN CALL PRTQLQ ;
172    3  3  |            IF QLQT | LQTRACE THEN DO;
             |               /* COUNT # UNCOVERED ELEMENTS
             |                  OF FINOSTAR & PRINT */
             |               REMUNCOV=0;
173    3  4  |               DO I=1 TO NE(K);
174    3  4  |                  IF FINOSTAR(I) THEN
175    3  5  |                     REMUNCOV=REMUNCOV+1;
             |               END;
176    3  5  |               PUT SKIP(2) EDIT(CVNOSTAR,
177    3  4  |               ' EVENTS NEVER COVERED BY A',
             |               ' PREVIOUS STAR ARE COVERED BY',
             |               ' THIS STAR, AND',REMUNCOV,
             |               ' REMAIN UNCOVERED')
             |               (F(4),A,A,A,F(4),A);
178    3  4  |            END;
179    3  3  |         END;
180    3  2  |      END;
             |
             |  /* FIND NEXT ELEMENT OF FINOMQ WHICH HAS NOT YET BEEN COVERED */
             |
181    3  1  |      DO IIL=1 TO NE(K);
182    3  2  |         IF FINOMQ(IIL) THEN DO;
183    3  3  |            CALL STAR ;
184    3  3  |            CALL BESTLQ ;
185    3  3  |            IF QLQT THEN CALL PRTQLQ ;
186    3  3  |            DELTA = DELTA + 1 ;
187    3  3  |         END;
188    3  2  |      END;
             |
             |        /* PRINT OUT CARTESIAN COMPLEXES COVERING CLASS K */
189    3  1  |        MQPTS=MQPTR-OMQPTR;
190    3  1  |        PUT SKIP(5) EDIT('THE FOLLOWING',MQPTS,
             |              ' CARTESIAN COMPLEXES COVER THE EVENTS IN CLASS',K,
             |              'COV  NEW  IND  TOT')
             |              (A,F(3),A,F(3),X(35),A);
191    3  1  |        IF SAVE THEN PUT FILE(COVER) SKIP EDIT(NV,NL,MQPTS,
             |              ' DELTA=',DELTA)((3)F(4),SKIP,A,F(4));
192    3  1  |        ISPL=1;    CALL TRIPLE;
194    3  1  |        DO I=OMQPTR+1 TO MQPTR;
```

```
STMT LEV NT

 195   3  2 |                     CALL PRCOVER(MQ.INT(I));
            |
 195   3  2 |                     ISPL=ISPL+1;
 197   3  2 |            IF SAVE THEN
            |            PUT FILE(COVER) SKIP LIST(MQ.INT(I));
 198   3  2 |            PUT SKIP;
 199   3  2 |              END;
 200   3  1 |              PUT SKIP(3) EDIT('**** DELTA FOR THIS SET IS',DELTA,
            |                     ' ****')(A,F(3),A);
 201   3  1 |              PUT SKIP(3) EDIT('THE LARGEST STAR HAD',LSS,
            |                     ' ELEMENTS; THE LARGEST INTERMEDIATE STAR HAD',
            |                     LNEAL,' ELEMENTS')(A,F(5),A,F(5),A);
 202   3  1 |              IF NUMSTRIM > 0 THEN PUT SKIP(3) EDIT
            |                     ('NOTE: INTERVALS RESULTING FROM TRIMMED STARS.',
            |                      ' TOTAL # STARS TRIMMED =',NUMSTRIM)(A,A,F(4));
            |
 203   3  1 |          OLPTR=OLPTR+1;
 204   3  1 |      END NEXTOEL;
            |
 205   3  0 | NTNTIC:PUT SKIP(3) LIST('*** NORMAL TERMINATION ***');
 206   3  0 |          GO TO NEWDATA;
            |
            |/* ****************************************************************
            |   ************************************************************ */
            |
 207   3  0 |STAR:    PROCEDURE REORDER ;
            |
            |          /*  THIS PROCEDURE DETERMINES THE STAR FOR ELEMENT IIL
            |          OF ARRAY E(K,*). UPON RETURN THE STAR WILL BE STORED
            |          BETWEEN POINTERS FGPTR AND LGPTR IN THE G LIST.      */
            |
 208   4  0 |                NUMTRIM = 0 ;
            |
            |                /* START FORMING G(E) BY (E MIX (COMPL(EL))) */
            |
 209   4  0 |                CALL MIX(E(K,IIL),FDOWN(1),INTP,POSP,NUM);
 210   4  0 |                IF STRACE THEN PUT SKIP(2) EDIT
            |                      ('START G(E)',(INTP(IG),POSP(IG)
            |                                           DO IG = 1 TO NUM)   )
            |                      (A,(NUM)(X(4),B,F(4)) ) ;
 211   4  0 |                IF NUM = 0 THEN
            |                      DO ;
 212   4  1 |                            FMIX = '1'B ;
 213   4  1 |                            GO TO TEOF ;
 214   4  1 |                      END ;
 215   4  0 |                DO IG = 1 TO NUM ;
 216   4  1 |                      CALL GETG(LGPTR) ;
 217   4  1 |                      IF IG = 1 THEN FGPTR = LGPTR ;
 218   4  1 |                      G.INT(LGPTR) = ONES ;
```

```
STMT LEV NT

 219   4  1  |                SUBSTR(G.INT(LGPTR),NVP(POSP(IG)),NLEV(POSP(IG)))
             |                      = INTP(IG);
 220   4  1  |                IF STRACE THEN DO;
 221   4  2  |                   PUT SKIP EDIT
             |                     ('FIRST INTERVAL:')(A);
 222   4  2  |                     CALL PRCOVER(G.INT(LGPTR));
 223   4  2  |                  END;
 224   4  1  |             END ;
 225   4  0  |             G.FPTR(LGPTR) = 0 ;
 226   4  0  |             G.BPTR(FGPTR) = 0 ;
 227   4  0  |             NES,LNES = NUM ;
 228   4  0  |             IF QST THEN PUT SKIP(2) EDIT
             |                  ('FINISHED MULT. FOR FOE(    1)',
             |                   'STAR SIZE =',NES,
             |                   'LARGEST INTERMEDIATE STAR SIZE =',LNES)
             |                  (A,X(8),A,F(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)) ...     */
             |
 229   4  0  |    MPROD:   DO IO = 2 TO NDOWN;
 230   4  1  |                  NES = 0 ;
 231   4  1  |                  CALL MIX(E(K,I1L),FDOWN(IO),INTP,POSP,NUM);
 232   4  1  |                  IF STRACE THEN PUT SKIP(2) EDIT
             |                        ('NEXT G(E)',(INTP(IG),POSP(IG)
             |                                     DO IG = 1 TO NUM) )
             |                        (A,(NUM)(X(4),B,F(4)) ) ;
 233   4  1  |                  IF NUM = 0 THEN
             |                        DO ;
 234   4  2  |                              FMIX = '0'B ;
 235   4  2  |                              GO TO TEQF ;
 236   4  2  |                        END ;
 237   4  1  |             KLC,KEC,KBC = 0;
 238   4  1  |                  NFGPTP = 0 ;    /* INITIALIZE NEXT FIRST G PTR   */
 239   4  1  |                  GPTR = FGPTR ; /* START LOOP OVER "I"    */
 240   4  1  |    ZEROM:        DO J = 1 TO NUM ;    /*  ZERO M ARRAY    */
 241   4  2  |                        M(J) = '0'B ;
 242   4  2  |                  END ;
             |
             |                  /*  NOW RUN THROUGH ALL INTERVALS RESULTING FROM
             |                      LAST CALL TO MIX...                        */
 243   4  1  |    OVERJ:        DO J = 1 TO NUM ;
 244   4  2  |                        TINT = SUBSTR(G.INT(GPTR),NVP(POSP(J)),
             |                                          NLEV(POSP(J)));
 245   4  2  |                        TAND = TINT & INTP(J) ;
             |                        /*  TEST FOR CONDITIONS 1 AND 3:     */
 246   4  2  |                        IF TAND = TINT THEN
             |                              DO ;   /*  CONDITION 1 OR 3, SO
```

```
STMT LEV NT

                                         STORE INTERVAL G(GPTR) INTO
                                          NEW G LIST.           */
 247   4  3                          CALL GETG(NLGPTR) ;
 248   4  3                          IF NFGPTR = 0 THEN
                                          DO ;
 249   4  4                                   NFGPTR = NLGPTR ;
 250   4  4                                   G.BPTR(NFGPTR) = 0 ;
 251   4  4                               END ;
 252   4  3                          G.INT(NLGPTR) = G.INT(GPTR) ;
 253   4  3                          NES = NES + 1 ;
 254   4  3                          IF NES > LNES THEN LNES = NES ;
 255   4  3                          KLC(J) = KLC(J) + 1 ;
 256   4  3                          KL(J,KLC(J)) = NLGPTR ;
 257   4  3                          CALL CHECKE(G.INT(GPTR)) ;
 258   4  3                          GO TO ENDI ;
 259   4  3                     END ;
 260   4  2                ELSE  /*  CHECK FOR CONDITION 2:    */
                             IF TAND = INTP(J) THEN
                                /*  CONDITION 2     */
                                M(J) = '1'B ;
 261   4  2           END OVERJ ;

                       /*  DO TERM BY TERM MULTIPLY        */
 262   4  1   TBTMULT:  DO J = 1 TO NUM ;
 263   4  2                TAND = INTP(J) & SUBSTR(G.INT(GPTR),
                                      NVP(POSP(J)),NLEV(POSP(J)));
 264   4  2                IF TAND = ZEROS THEN GO TO ENDTMULT ;
 265   4  2                TAINT = G.INT(GPTR) ;
 266   4  2                SUBSTR(TAINT,NVP(POSP(J)),NLEV(POSP(J)))=TAND;

                       /*  CHECK TO SEE IF ANY OF THE PREVIOUS
                           TERMS GENERATED OUT OF THE JTH INTERVAL
                           COVERS THIS NEW INTERVAL          */

 267   4  2                DO IKL = 1 TO KLC(J) ;
 268   4  3                     IF (G.INT(KL(J,IKL)) & TAINT) = TAINT
                                     THEN GO TO ENDTMULT ;
 269   4  3                END ;
 270   4  2                IF ¬M(J) THEN   /*  CHECK THIS CONDITION 4
                                             INTERVAL TO SEE IF IT IS
                                             COVERED BY A PREVIOUS
                                             CONDITION 4 INTERVAL    */
                             DO IKL = 1 TO KBC(J) ;
 271   4  3                     IF (G.INT(KB(J,IKL)) & TAINT) = TAINT
                                     THEN GO TO ENDTMULT ;
 272   4  3                  END ;

                       /*  PUT INTERVAL INTO NEW G(E)     */
 273   4  2                CALL GETG(NLGPTR) ;
```

```
STMT LEV NT

 274    4  2          IF NFGPTR = 0 THEN
                           DO ;
 275    4  3                    NFGPTR = NLGPTR ;
 276    4  3                    G.BPTR(NFGPTR) = 0 ;
 277    4  3               END ;
 278    4  2          G.INT(NLGPTR) = TAINT ;
 279    4  2          NES = NES + 1 ;
 280    4  2          IF NES > LNES THEN LNES = NES ;
 281    4  2          IF STRACE THEN                        DO;
 282    4  3    PUT SKIP(2) EDIT
                                   ('NEW INTERVAL:')(A);
 283    4  3                       CALL PRCOVER(TAINT); END;

 285    4  2          IF M(J) THEN
                           DO ;
                                /*  CHECK TO SEE IF THIS NEW INTERVAL
                                COVERS ANY PREVIOUS TERMS GENERATED
                                OUT OF THE JTH INTERVAL        */
 286    4  3                    CALL CHECKE(TAINT) ;
                                /* PUT IN KL POINTER AS THIS ONE
                                   COULD COVER LATER INTERVALS   */
 287    4  3                    KLC(J) = KLC(J) + 1 ;
 288    4  3                    KL(J,KLC(J)) = NLGPTR ;
                                /* PUT IN KE POINTER FOR THIS ONE
                                   AS IT CAN BE COVERED LATER   */
 289    4  3                    KEC(J) = KEC(J) + 1 ;
 290    4  3                    KE(J,KEC(J)) = NLGPTR ;
 291    4  3               END ;
 292    4  2               ELSE DO ;  /* CHECK CONDX 4 COVERAGE   */
 293    4  3                    CALL CKBE(TAINT) ;
                                /*  PUT IN KB POINTER AS THIS
                                INTERVAL IS A CONDX 4 INTERVAL   */
 294    4  3                    KBC(J) = KBC(J) + 1 ;
 295    4  3                    KB(J,KBC(J)) = NLGPTR ;
 296    4  3               END ;

 297    4  2  ENDTMULT:  END TBTMULT ;

 298    4  1  ENDI:      GPTR = G.FPTR(GPTR) ;  /*  "INCREMENT" "I"   */
 299    4  1             IF GPTR ¬= 0 THEN GO TO ZEROM ;   /*  TEST FOR END
                                                               OF LOOP   */
 300    4  1             CALL RETURNG(FGPTR,LGPTR) ;
 301    4  1             FGPTR = NFGPTR ;             /*  UPDATE POINTERS   */
 302    4  1             LGPTR = NLGPTR ;
 303    4  1             G.FPTR(LGPTR) = 0 ;
 304    4  1             IF STRACE THEN CALL PRTSTAR ;
 305    4  1             IF QST THEN PUT SKIP(2) EDIT
                              ('FINISHED MULT. FOR FOE(',IO,')',
                               'STAR SIZE =',NES,
```

```
STMT LEV NT

                 |                        'LARGEST INTERMEDIATE STAR SIZE =',LNES)
                 |                      (A,F(4),A,X(8),A,F(5),X(4),A,F(5)) ;
306    4   1     |                  IF (NES > MAXSTAR) & (IO < NDOWN)
                 |                     THEN CALL TRIMSTAR;
                 |
307    4   1     |             END MPROD ;
                 |
308    4   0     |             IF LNES > LNEAL THEN LNEAL = LNES ;
309    4   0     |             IF NES  > LSS   THEN LSS   = NES  ;
310    4   0     |             IF NUMTRIM > 0 THEN NUMSTRIM = NUMSTRIM + 1 ;
311    4   0     |         RETURN ;
                 |
312    4   0     |   TEOF: DO I=1 TO NV;
313    4   1     |              KBC(I)=INDEX(SUBSTR(E(K,I1L),NVP(I),NLEV(I)),'1'B) - 1;
314    4   1     |         END;
315    4   0     |         PUT SKIP(3) EDIT
                 |              ('*** DATA ERROR: TRUE EVENT = FALSE EVENT =',KBC)
                 |               (A,(NV)F(6)) ;
316    4   0     |         IF FMIX THEN GO TO NEWDATA ;
317    4   0     |         CALL RETURNG(FGPTR,LGPTR) ;
318    4   0     |         GO TO NEWDATA ;
                 |
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 FINDMQ.                                          */
                 |
321    4   0     |         DECLARE (CPTR,CNLT) (CUTSTAR) FIXED BINARY(15),
                 |                 (MINPTR,MINI,NLIT,GPTR,I)
                 |                      FIXED BINARY(15) STATIC ,
                 |                  (MINWT,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
```

```
STMT LEV NT

                                   THE ORIGINAL STAR ELEMENTS
                          I      - INDEX USED FOR INDEXING THROUGH CUT STAR
                          MINI   - 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
                                                                          */

 322   4  0           GPTR = FGPTR ;    /*  SET POINTER TO FIRST ELEMENT OF STAR */
 323   4  0           MINWT = 50000 ;
 324   4  0           DO I = 1 TO CUTSTAR ;   /* GET INITIAL ELEMENTS FOR CUT */
 325   4  1              CPTR(I) = GPTR ;   /* STAR BY TAKING FIRST CUTSTAR */
 326   4  1              CALL LCOVER(GPTR,WT) ;   /* ELEMENTS OF STAR     */
 327   4  1              CWT(I) = WT ;
 328   4  1              NLIT=NUMLIT(GPTR);
 329   4  1              CNLT(I) = NLIT ;
 330   4  1              IF WT > MINWT THEN GO TO INCR_PTR ;
 331   4  1              IF WT = MINWT THEN
                            IF NLIT <= CNLT(MINI) THEN GO TO INCR_PTR ;
 332   4  1              MINWT = WT ;
 333   4  1              MINI = I ;
 334   4  1        INCR_PTR:
                         GPTR = G.FPTR(GPTR) ;
 335   4  1           END ;
 336   4  0     ELIMSMALL:
                      CALL LCOVER(GPTR,WT) ;
 337   4  0           IF WT < MINWT THEN  /* REMOVE THIS ELEMENT FROM OLD STAR */
                         GO TO REMOVE_FROM_OLD_STAR ;
 338   4  0           NLIT=NUMLIT(GPTR);
 339   4  0           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   */
 340   4  0           MINPTR = CPTR(MINI) ;
 341   4  0           CNLT(MINI) = NLIT ;
 342   4  0           CPTR(MINI) = GPTR ;
 343   4  0           CWT(MINI)  = WT   ;
 344   4  0           IF G.FPTR(GPTR) ¬= 0 THEN  /*  FIND MINIMUM WEIGHT    */
                         DO ;
 345   4  1                 MINWT = 50000 ;
 346   4  1                 DO I = 1 TO CUTSTAR ;
 347   4  2                    IF CWT(I) > MINWT THEN GO TO NO_EXCHANGE ;
 348   4  2                    IF CWT(I) = MINWT THEN
                                  IF CNLT(MINI) >= CNLT(I) THEN
                                     GO TO NO_EXCHANGE ;
 349   4  2                    MINWT = CWT(I) ;
```

```
STMT LEV NT

 350   4  2  |                      MINI  = I  ;                                   |
 351   4  2  |     NO_EXCHANGE: END ;                                               |
 352   4  1  |              END ;                                                   |
 353   4  0  |        GO TO INCR_I ;                                                |
 354   4  0  |REMOVE_FROM_OLD_STAR:                                                 |
             |        MINPTR = GPTR ;                                               |
 355   4  0  |INCR_I: GPTR = G.FPTR(GPTR) ;                                         |
 356   4  0  |        CALL REMOVEG(MINPTR) ;                                        |
 357   4  0  |        IF GPTR ¬= 0 THEN GO TO ELIMSMALL ; /* END OF STAR? */        |
 358   4  0  |        NES = CUTSTAR ;    /* YES, CLEANUP AND RETURN      */         |
 359   4  0  |        IF QST | STRACE THEN PUT SKIP(2) LIST                         |
             |             ('***  STAR REDUCED TO CUTSTAR SIZE   ***') ;            |
 360   4  0  |        IF STRACE THEN CALL PRTSTAR ;                                 |
 361   4  0  |        NUMTRIM = NUMTRIM + 1 ;                                       |
 362   4  0  |        RETURN ;                                                      |
             |                                                                      |
 363   4  0  |    END TRIMSTAR ;                                                    |
             |                                                                      |
             |/* *************************************************************      |
             |   ************************************************************* */  |
             |                                                                      |
 364   3  0  |NUMLIT:   PROCEDURE(GPTR) RETURNS(FIXED BINARY(15)) REORDER;          |
             |                                                                      |
             |        /*  THIS PROCEDURE DETERMINES ALIT, THE NUMBER OF LITERALS IN |
             |        THE INTERVAL OF G(E) POINTED TO BY GPTR.                 */   |
             |                                                                      |
 365   4  0  |        DECLARE (GPTR,ALIT) FIXED BINARY(15),                         |
             |                 LIT FIXED BINARY(15) STATIC ;                        |
             |                                                                      |
 366   4  0  |        ALIT = 0 ;                                                    |
 367   4  0  |        DO LIT = 1 TO NV ;                                            |
 368   4  1  |              IF SUBSTR(G.INT(GPTR),NVP(LIT),NLEV(LIT))               |
             |                    ¬= SUBSTR(ONES,1,NLEV(LIT)) THEN ALIT=ALIT+1;     |
 369   4  1  |        END ;                                                         |
 370   4  0  |        RETURN(ALIT);                                                 |
             |                                                                      |
 371   4  0  |    END NUMLIT ;                                                      |
             |      /* ***********************************************************  |
             |    ************************************************************ */  |
             |                                                                      |
 372   3  0  |LCOVER: PROCEDURE(L,NFIL) REORDER ;                                   |
             |                                                                      |
             |        /*  THIS PROCEDURE DETERMINES NFIL, THE NUMBER OF             |
             |        ELEMENTS OF THE CURRENT FINDMQ THAT ARE COVERED BY THE INTERVAL|
             |        IN G(E) POINTED TO BY THE POINTER L                 */        |
             |                                                                      |
 373   4  0  |        DECLARE (I1 STATIC,L) FIXED BINARY(15) ,                      |
             |                 NFIL FIXED BIN(31);                                  |
             |                                                                      |
```

```
STMT LEV NT

 374   4  0 |        NF1L = 0 ;
 375   4  0 |        DO I1 = 1 TO NE(K);
 376   4  1 |             IF F1NOMQ(I1) & EINCPLX(I1,L) THEN NF1L = NF1L + 1 ;
 377   4  1 |        END ;
              |
 378   4  0 |        RETURN ;
              |
 379   4  0 |    END LCOVER ;
              |
              |/* ******************************************************************:
              |  ************************************************************** */
              |
              |
 380   3  0 |  BESTLQ: PROCEDURE REORDER;
              |
              |        /* 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 (CRITVAL) IS DETERMINED FOR EVERY
              |           COMPLEX IN THE STAR. THOSE COMPLEXES WHOSE CRITVAL LIES
              |           WITHIN A RANGE OF THE MINIMUM CRITVAL 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 CRITVALS. WHENEVER A SINGLE
              |           COMPLEX REMAINS, THE ROUTINE RETURNS IMMEDIATELY WITH
              |           ITS LQ-POINTER. OTHERWISE, WHEN ALL CRITERIA
              |           ARE PROCESSED, THE FIRST COMPLEX WITH MINIMUM CRITVAL
              |           IS CHOSEN. THE COMPLEX IS ADDED TO THE CURRENT MQ-LIST
              |           AND ALL APPROPRIATE FLAGS & COUNTS ARE UPDATED. */
              |
 381   4  0 |     DECLARE   (ICRIT,CRITX,LPTR,NUMCPLX,A,B,LASTPOS)
              |                      FIXED BINARY(15),
              |               (UBOUND,MAXVAL,MINVAL, V ) DEC FLOAT;
              |
 382   4  0 |        NUMCPLX,CVNOSTAR=0; LPTR=FGPTR;
 384   4  0 |        DO WHILE (LPTR¬=0);     /* COUNT # COMPLEXES IN STAR AND
              |                                  SET FINOSTAR TO '0'B FOR ALL
              |                                  EVENTS IN STAR */
              |
 385   4  1 |             NUMCPLX=NUMCPLX+1;
 386   4  1 |             DO I=1 TO NE(K);
 387   4  2 |                  IF FINOSTAR(I) & EINCPLX(I,LPTR) THEN DO;
 388   4  3 |                       FINOSTAR(I)='0'B; CVNOSTAR=CVNOSTAR+1;
 390   4  3 |                  END;
 391   4  2 |             END;
 392   4  1 |             LPTR=G.FPTR(LPTR);
 393   4  1 |        END;
              |
 394   4  0 |        IF LQTRACE THEN DO;
 395   4  1 |             PUT PAGE EDIT ('STAR FOR TRUE EVENT',I1L,' IN CLASS',K)
```

```
STMT LEV NT

                  |              (A,F(4),A,F(3) );
 396    4   1     |          LPTR=FGPTR;
 397    4   1     |          DO WHILE(LPTR¬=0);
 398    4   2     |              CALL PRCOVER(G.INT(LPTR));
 399    4   2     |              PUT SKIP LIST('CRIT     VALUE');
 400    4   2     |              DO ICRIT=1 TO NCRIT;
 401    4   3     |                  CRITX=CLIST(ICRIT);
 402    4   3     |                  PUT SKIP EDIT(CRITX,ABS(CRITVAL(CRITX)))
                  |                     (F(3),F(10,2));
 403    4   3     |              END;
 404    4   2     |              LPTR=G.FPTR(LPTR);
 405    4   2     |          END;
 406    4   1     |          PUT SKIP(2) EDIT ('LARGEST INTERMEDIATE STAR SIZE WAS',
                  |              LNES)(A,F(5));
 407    4   1     |      END;
                  |
 408    4   0     | CRITLOOP:
                  |      DO ICRIT=1 TO NCRIT;
 409    4   1     |          MINVAL=1.E60;
 410    4   1     |          IF NUMCPLX=1 THEN GOTO CHOOSE;
                  |
 411    4   1     |          MAXVAL=-MINVAL;
 412    4   1     |          CRITX=CLIST(ICRIT);
 413    4   1     |          LPTR=FGPTR;
 414    4   1     |        LQLOOP: DO WHILE (LPTR¬=0);
                  |              /* CALCULATE CRITVAL FOR GIVEN LQ */
 415    4   2     |              V, G.VAL(LPTR)=CRITVAL(CRITX);
 416    4   2     |              IF MAXVAL<V THEN MAXVAL=V;
 417    4   2     |              IF MINVAL>V THEN MINVAL=V;
 418    4   2     |              LPTR=G.FPTR(LPTR);
                  |
 419    4   2     |        END LQLOOP;
 420    4   1     |          UBOUND=MINVAL+TLIST(ICRIT)*(MAXVAL-MINVAL);
 421    4   1     |          LPTR=FGPTR;
 422    4   1     |          DO WHILE (LPTR¬=0);
 423    4   2     |              IF G.VAL(LPTR) > UBOUND THEN DO;
                  |                  /* DELETE LQ'S WHICH DON'T MEET CRITERIA */
 424    4   3     |                  I=G.FPTR(LPTR);
 425    4   3     |                  CALL REMOVEG(LPTR);
 426    4   3     |                  LPTR=I;
 427    4   3     |                  NUMCPLX=NUMCPLX-1;
 428    4   3     |              END;
 429    4   2     |             ELSE LPTR=G.FPTR(LPTR);
 430    4   2     |          END;
 431    4   1     |  END CRITLOOP;
                  |
                  |      /* CHOOSE COMPLEX WITH MINIMUM CRITVAL */
 432    4   0     | CHOOSE: LQ=FGPTR;
 433    4   0     |      DO I=1 TO NUMCPLX WHILE (MINVAL < G.VAL(LQ));
```

```
STMT LEV NT

 434   4  1  |              LQ=G.FPTR(LQ);
 435   4  1  |        END;
             |
 436   4  0  |        IF LQTRACE THEN DO;
 437   4  1  |              PUT SKIP(5) LIST('LQ SELECTED FOR THIS EVENT IS:');
 438   4  1  |              CALL PRCOVER(G.INT(LQ));
 439   4  1  |        END;
             |
 440   4  0  |        IF LQST='1'B THEN DO;
             |              /* DETERMINE LQ-STAR FROM CURRENT LQ
             |                 I.E. OPTIMIZE CHOSEN COVER */
 441   4  1  |              TAINT=ZEROS;
 442   4  1  |              DO I=1 TO NE(K);
 443   4  2  |                    IF FINOMQ(I) & EINCPLX(I,LQ) THEN
             |                          TAINT =TAINT | E(K,I);
 444   4  2  |              END;
 445   4  1  |              DO I=1 TO NV;
 446   4  2  |                    IF SUBSTR(G.INT(LQ),NVP(I),NLEV(I))
             |                          = SUBSTR(ONES,1,NLEV(I)) THEN
             |                            SUBSTR(TAINT,NVP(I),NLEV(I))=ONES;
 447   4  2  |                    IF ¬TYPCOV(I) THEN DO;
             |                          /* CONVERT FACTORS INTO INTERVALS */
 448   4  3  |                          LASTPOS=NVP(I)+NLEV(I)-1;
 449   4  3  |                          DO A=NVP(I) TO LASTPOS
             |                             WHILE(¬SUBSTR(TAINT,A,1));
 450   4  4  |                          END; /* A = POS. OF 1ST '1' IN VARIABLE */
 451   4  3  |                          DO B=LASTPOS TO NVP(I) BY -1
             |                             WHILE (¬SUBSTR(TAINT,B,1));
 452   4  4  |                          END; /* B = POS. OF LAST '1' IN VARIABLE */
 453   4  3  |                          SUBSTR(TAINT,A,B-A+1)=ONES;
 454   4  3  |                    END;
 455   4  2  |              END;
 456   4  1  |              G.INT(LQ)=TAINT;
 457   4  1  |        END;
             |
             |        /* ADD CHOSEN LQ TO CURRENT MQ-LIST */
 458   4  0  |        MQPTR=MQPTR+1;
 459   4  0  |        IF MQPTR > NMQ THEN DO;
 460   4  1  |              PUT SKIP(3) LIST('*** MQ SPACE EXCEEDED ***');
 461   4  1  |              CALL RETURNG(FGPTR,LGPTR);
 462   4  1  |              GOTO NEWDATA;
 463   4  1  |        END;
 464   4  0  |        MQ(MQPTR) = G(LQ), BY NAME;
 465   4  0  |        IF NUMTRIM > 0 THEN MQ.TRIM(MQPTR) = '#';
 466   4  0  |                 ELSE MQ.TRIM(MQPTR)= ' ';
             |
             |        /* RESET PROPER ELEMENTS OF FINOMQ & COUNT
             |           NO. NEW EVENTS COVERED */
 467   4  0  |        NFILQ=0;    COMCOVD=0;
```

```
STMT LEV NT

 469   4  0  |      DO I=1 TO NE(K);
 470   4  1  |            IF FINOMQ(I) & EINCPLX(I,LQ) THEN DO;
 471   4  2  |                  FINOMQ(I)='0'B; NFILQ=NFILQ+1;
 473   4  2  |            END;
 474   4  1  |            IF EINCPLX(I,LQ) THEN DO;
 475   4  2  |                  COMCOVD=COMCOVD+1;
 476   4  2  |            END;
 477   4  1  |      END;
 478   4  0  |      NOFILQ=NOFILQ+NFILQ;
 479   4  0  |     TOTCOV(ISPL)=NOFILQ;
 480   4  0  |     NEWCOV(ISPL)=NFILQ;
 481   4  0  |     NUMCOV(ISPL)=COMCOVD;
 482   4  0  |     ISPL=ISPL+1;
             |
 483   4  0  |      IF LQTRACE THEN DO;
 484   4  1  |            PUT SKIP(2) EDIT('THIS LQ COVERS',NFILQ,
             |                  ' NEW EVENTS AND A TOTAL OF',NOFILQ,
             |                  ' EVENTS ARE NOW COVERED IN MQ')
             |                  (A,F(4),A,F(4),A);
 485   4  1  |            PUT SKIP(2) LIST('CONDITION OF FINOSTAR IS NOW');
 486   4  1  |            PUT SKIP DATA((FINOSTAR(I)DO I=1 TO NE(K)));
 487   4  1  |            PUT SKIP(2) LIST('CONDITION OF FINOMQ IS NOW');
 488   4  1  |            PUT SKIP DATA((FINOMQ(I)DO I=1 TO NE(K)));
 489   4  1  |            IF ¬QLQT & NUMTRIM>0 THEN PUT SKIP(2) EDIT
             |                  ('*** STAR WAS TRIMMED',NUMTRIM,' TIMES ***')
             |                  (A,F(4),A);
 490   4  1  |      END;
             |
 491   4  0  |      CALL RETURNG(FGPTR,LGPTR);
 492   4  0  |      RETURN;
             |
             | /*-------------------------------------------------------------*/
 493   4  0  | CRITVAL:  PROCEDURE(CRITX) RETURNS(DEC FLOAT) REORDER;
 494   5  0  |      DECLARE (I,CRITX) FIXED BINARY(15), CRIT(7) LABEL,
             |            VI DEC FLOAT STATIC, STRING BIT(NLMAX) VARYING;
             |
 495   5  0  |      TAINT=G.INT(LPTR);
 496   5  0  |      V, VI = 0.;
 497   5  0  |      GO TO CRIT(CRITX);
             |
 498   5  0  | CRIT(1):     /* MAXIMIZE # OF EVENTS NOT COVERED BY MQ */
             |      DO I=1 TO NE(K);
 499   5  1  |            IF FINOMQ(I) & EINCPLX(I,LPTR) THEN V=V+1.;
 500   5  1  |      END;
 501   5  0  |      RETURN(-V);
             |
 502   5  0  | CRIT(2):     /* MINIMIZE # VARIABLES (LITERALS) */
             |      RETURN(FLOAT(NUMLIT(LPTR)));
             |
```

```
STMT LEV NT

 503    5  0  | CRIT(3):      /* MINIMIZE SUM(COST-PER-VARIABLE) */
              |        DO I=1 TO NV;
 504    5  1  |               IF SUBSTR(TAINT,NVP(I),NLEV(I))
              |                     ¬= SUBSTR(ONES,1,NLEV(I)) THEN V=V+Z(I);
 505    5  1  |        END;
 506    5  0  |        RETURN(V);
              |
 507    5  0  | CRIT(4):      /* MINIMIZE GENERALIZATION COEFFICIENT */
              |        V = 1.;
 508    5  0  |        DO I=1 TO NV;      /* FIND TOTAL # CELLS IN COMPLEX */
 509    5  1  |               V1 = RANGE(I);
 510    5  1  |               IF V1=0. THEN V1=NLEV(I);
 511    5  1  |               V = V*V1;
 512    5  1  |        END;
 513    5  0  |        V1 = 0.;
 514    5  0  |        DO I=1 TO NE(K);   /* FIND # CELLS IN THIS CLASS */
 515    5  1  |               IF EINCPLX(I,LPTR) THEN V1=V1+1.;
 516    5  1  |        END;
 517    5  0  |        RETURN(V/V1);
              |
 518    5  0  | CRIT(5):      /* MAXIMIZE SUM(WEIGHTS OF EVENTS NOT IN MQ) */
              |        DO I=1 TO NE(K);
 519    5  1  |               IF FINOMQ(I) & EINCPLX(I,LPTR) THEN V=V+W(K,I);
 520    5  1  |        END;
 521    5  0  |        RETURN(-V);
              |
 522    5  0  | 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;
 523    5  1  |               IF CRITX=6 THEN V=V+RANGE(I);
 524    5  1  |                          ELSE V=V+VARIANCE(I);
 525    5  1  |        END;
 526    5  0  |        RETURN(V);
              |
 527    5  0  | RANGE:  PROCEDURE(IVAR) RETURNS(DEC FLOAT) REORDER;
 528    6  0  |        DECLARE (IND,IVAR) FIXED BINARY(15), RG DEC FLOAT;
              |
 529    6  0  |        RG = 0.;
 530    6  0  |        STRING = SUBSTR(TAINT,NVP(IVAR),NLEV(IVAR));
 531    6  0  |        IF STRING=SUBSTR(ONES,1,NLEV(IVAR)) THEN RETURN(RG);
 532    6  0  |      CHKIND:  IND= INDEX(STRING,'1'B);
 533    6  0  |        IF IND=0 THEN RETURN(RG);
 534    6  0  |        RG=RG+1.;
 535    6  0  |        SUBSTR(STRING,IND,1) = '0'B;
 536    6  0  |        GO TO CHKIND;
 537    6  0  |        END RANGE;
              |
 538    5  0  | VARIANCE:  PROCEDURE(IVAR) RETURNS(DEC FLOAT) REORDER;
 539    6  0  |        DECLARE (IVAR,J,NLEV,LEV(1:NLEV(IVAR))) FIXED BINARY(15),
```

```
STMT LEV NT

                   |                (AVG,VCE) DEC FLOAT STATIC;
                   |
 540   6  0        |          VCE = 0.;
 541   6  0        |          STRING = SUBSTR(TAINT,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,1) THEN DO;
 547   6  2        |                      NLEV=NLEV+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 CRITVAL;
                   |
                   |   /*----------------------------------------------------------*/
                   |
                   |
 558   4  0        |   END BESTLQ;
                   |
                   |/* ****************************************************************
                   |    ************************************************************ */
                   |
 559   3  0        |     PRTSTAR: PROCEDURE REORDER ;
                   |
                   |          /*  THIS PROCEDURE PRINTS OUT THE CONTENTS OF G(E)
                   |          BETWEEN POINTERS FGPTR AND THE ELEMENT OF G(E) WHICH
                   |          HAS A ZERO FORWARD POINTER (G.FPTR(GPTR) = 0 )   */
                   |
 560   4  0        |          DECLARE GPTR FIXED BINARY(15) STATIC;
                   |
 561   4  0        |          GPTR = FGPTR ;
 562   4  0        |          DO WHILE (GPTR ¬= 0);
 563   4  1        |                CALL PRCOVER(G.INT(GPTR));
 564   4  1        |                GPTR = G.FPTR(GPTR) ;
 565   4  1        |          END;
 566   4  0        |          RETURN ;
                   |
 567   4  0        |     END PRTSTAR ;
                   |
                   |/* *****************************************************************
                   |    ************************************************************* */
```

```
STMT LEV NT

568   3  0  |PRTQLQ: PROCEDURE REORDER ;
            |
            |        /*  THIS PROCEDURE PRINTS OUT ABBREVIATED INFORMATION
            |        ABOUT THE NEWLY OBTAINED STAR AND THE ASSOCIATED LQ.   */
            |
569   4  0  |        PUT SKIP(3) EDIT(REPEAT('-',100))(A);
570   4  0  |        PUT SKIP(1) EDIT
            |             ('STAR',MQPTR,'  COVERING EVENT',I1L,'  HAS',NES,
            |              ' COMPLEXES; LARGEST INTERMEDIATE STAR SIZE WAS',LNES)
            |             (A,F(4),A,F(4),A,F(5),A,F(5)) ;
571   4  0  |        IF NUMTRIM > 0 THEN PUT SKIP(2) EDIT
            |             ('***  NUMBER OF TIMES STAR WAS TRIMMED =',NUMTRIM,
            |              '  ***')   (A,F(4),A) ;
572   4  0  |         IF SAVELQ THEN PUT FILE(COVER) LIST(MQ.INT(MQPTR));
573   4  0  |         CALL PRCOVER(MQ.INT(MQPTR));
574   4  0  |        PUT SKIP(2) EDIT
            |             ('THIS LQ COVERS',NF1LQ,'NEW EVENTS AND A TOTAL OF',
            |              NOF1LQ,' EVENTS ARE NOW COVERED IN MQ')
            |              (A,F(4),A,F(4),A);
575   4  0  |        RETURN ;
            |
576   4  0  |    END PRTQLQ ;
            |
            |/* ****************************************************************
            |   ************************************************************* */
            |
577   3  0  |READVEC:     PROCEDURE REORDER ;
            |
            | /* THIS PROC READS IN THE EVENTS ASSOCIATED WITH THE CLASSES
            |    E(0,*),....,E(MAXCL,*) AND CONVERTS THEM TO BI-UNARY STRINGS.
            |    INPUT EVENTS ARE ASSUMED TO BE IN VECTOR FORM. */
            |
578   4  0  |        DECLARE (I,J,K) FIXED BINARY(15);
            |
579   4  0  |        DO K=0 TO MAXCL;
580   4  1  |              PUT SKIP(2) EDIT('CLASS F(',K,')')(A,F(2),A);
581   4  1  |              DO I=1 TO NE(K);
582   4  2  |                    GET LIST(POSP);
583   4  2  |                    PUT SKIP EDIT('EVENT NO.',I,'=',POSP)
            |                          (X(5),A,F(4),A,(NV)F(6));
584   4  2  |                    E(K,I)=ZEROS;
585   4  2  |                    DO J=1 TO NV;
586   4  3  |                          IF POSP(J)<0 | POSP(J)>NLEV(J)-1 THEN DO;
587   4  4  |                                PUT SKIP(2) EDIT('*** VARIABLE',
            |                                     J,' INCORRECTLY SPECIFIED',
            |                                     ' IN ABOVE EVENT ***')(A,F(3),A,A);
588   4  4  |                                GO TO NEWDATA;
589   4  4  |                          END;
```

```
STMT LEV NT

590   4  3  |                          SUBSTR(E(K,I),NVP(J)+POSP(J),1) = '1'B;
591   4  3  |                    END;
592   4  2  |               END;
593   4  1  |          END READVEC;
            |
            |/* ******************************************************************
            |   ************************************************************** */
            |
594   3  0  |READGAM:      PROCEDURE REORDER ;
            |
            |/* THIS PROC READS IN THE EVENTS ASSOCIATED WITH THE CLASSES
            |    E(0,*),...,E(MAXCL,*) AND CONVERTS THEM TO BI-UNARY STRINGS.
            |          INPUT EVENTS ARE ASSUMED TO BE IN GAMMA FORMAT. */
            |
595   4  0  |          DECLARE (I,II,K,BVAL)FIXED BINARY(15),
            |                J  FIXED BINARY(31);
            |
596   4  0  |          DO K=0 TO MAXCL;
597   4  1  |                PUT SKIP(2) EDIT('CLASS F(',K,')')(A,F(2),A);
598   4  1  |                DO I=1 TO NE(K);
599   4  2  |                      GET LIST(J);
600   4  2  |                      PUT SKIP EDIT('EVENT NO.',I,'=',J)
            |                            (X(5),A,F(4),A,F(6));
601   4  2  |                      E(K,I)=ZEROS;
602   4  2  |                      DO II=NV TO 1 BY -1;
603   4  3  |                            BVAL=MOD(J,NLEV(II));
604   4  3  |                            J=J/NLEV(II);
605   4  3  |                            SUBSTR(E(K,I),NVP(II)+BVAL,1) = '1'B;
606   4  3  |                      END;
607   4  2  |                END;
608   4  1  | END READGAM;
            |
            |/* **************************************************************
            |   ************************************************************** */
            |
609   3  0  |   MIX:      PROCEDURE(E1,E0,IM,PM,NM) REORDER ;
            |
            |          /*  THIS PROCEDURE PERFORMS THE OPERATION E1 MIX (E0 COMPLEMENT
            |          THE RESULT IS STORED IN AN ABBREVIATED FORM IN ARRAYS IM AND PM
            |
            |          IM(I) CONTAINS THE BOUND PAIR OF THE LITERAL
            |          PM(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.                    */
            |
610   4  0  |          DECLARE ((E1,E0) BIT(*),IM(*) BIT(*)) ALIGNED,
            |                PM(*) FIXED BINARY(15),
```

```
STMT LEV NT

                     (((IP,J,K,BITPOS) STATIC),NM) FIXED BINARY(15),
                     (PARTEO,PARTE1) BIT(NLMAX) ALIGNED ;

 611   4  0      NM = 0 ;
 612   4  0      DO IP = 1 TO NV ;
 613   4  1           PARTEO = SUBSTR(EO,NVP(IP),NLEV(IP)) ;
 614   4  1           PARTE1 = SUBSTR(E1,NVP(IP),NLEV(IP)) ;
 615   4  1           IF PARTEO =(PARTE1 | PARTEO)THEN GO TO END_THIS_POSITION;
 616   4  1           NM = NM + 1 ;
 617   4  1           PM(NM) = IP ;
 618   4  1           IF TYPCOV(IP)     THEN            DO;
 619   4  2              IM(NM)=SUBSTR(¬PARTEO,1,NLEV(IP));
 620   4  2              GOTO END_THIS_POSITION;       END;
 622   4  1      BITPOS=INDEX(PARTE1,'1'B);    IM(NM)='0'B;
 624   4  1      DO J=BITPOS+1 TO NLEV(IP) WHILE(¬SUBSTR(PARTEO,J,1)); END;
 626   4  1      DO K=BITPOS-1 TO 1 BY -1 WHILE(¬SUBSTR(PARTEO,K,1)) ;END;
 628   4  1      SUBSTR(IM(NM),K+1,J-K-1)=ONES;

 629   4  1  END_THIS_POSITION:
                 END ;
 630   4  0      RETURN ;

 631   4  0  END MIX ;

            /* **************************************************************
              ************************************************************ */

 632   3  0   CHECKE:  PROCEDURE(BP) REORDER ;

                 /*  THIS PROCEDURE CHECKS EARLIER INTERVALS IN G(E) FOR
                 POSSIBLE COVERAGE BY THE INTERVAL REPRESENTED BY THE BOUND
                 PAIR BP.  ANY PREVIOUS INTERVALS COVERED ARE THROWN OUT
                 OF G(E) AND ITS ENTRY IS DELETED FROM THE KE LIST.  */

 633   4  0      DECLARE BP BIT(*) ALIGNED,
                       (THROWE,THROWL) BIT(1) ALIGNED STATIC,
                       (I,IKE,IKL,P) FIXED BINARY(15) STATIC ;

 634   4  0      THROWE = '0'B ;
 635   4  0      THROWL = '0'B ;
 636   4  0      DO IKE = 1 TO KEC(J) ;
 637   4  1           P = KE(J,IKE) ;
 638   4  1           IF (BP & G.INT(P)) = G.INT(P) THEN
                             /*  NEW INTERVAL COVERS OLD ONE;
                                 THROW OUT OLD ONE.           */
                         DO ;
 639   4  2                    NES = NES - 1 ;
 640   4  2                    CALL REMOVEGN(P) ;
 641   4  2                    I = KE(J,IKE) ;
```

```
STMT LEV NT

                |                       /*  SEE IF THIS POINTER IS IN KL LIST */
 642    4  2    |                       DO IKL = 1 TO KLC(J) ;
 643    4  3    |                            IF KL(J,IKL) = I THEN /* YES, REMOVE
                |                                                             IT   */
                |                                 DO ;
 644    4  4    |                                      THROWL = '1'B;
 645    4  4    |                                      KL(J,IKL) = 0 ;
 646    4  4    |                                      KLC(J) = KLC(J) - 1 ;
 647    4  4    |                                      GO TO REMOVEE ;
 648    4  4    |                                 END ;
 649    4  3    |                       END ;
                |                       /*  REMOVE POINTER FROM KE LIST    */
 650    4  2    |            REMOVEE:   KEC(J) = KEC(J) - 1 ;
 651    4  2    |                       KE(J,IKE) = 0 ;
 652    4  2    |                       THROWE = '1'B ;
 653    4  2    |                  END ;
 654    4  1    |         END ;
                |
 655    4  0    |         CALL CKBE(BP) ;
                |
 656    4  0    |         IF THROWE THEN  /*  COLLAPSE KE LIST     */
                |              DO ;
 657    4  1    |                  I = 0 ;
 658    4  1    |                  DO IKE = 1 TO KEC(J) ;
 659    4  2    |             INCRE:    I = I + 1 ;
 660    4  2    |                       IF KE(J,I) = 0 THEN GO TO INCRE ;
 661    4  2    |                       KE(J,IKE) = KE(J,I) ;
 662    4  2    |                  END ;
 663    4  1    |              END ;
 664    4  0    |         IF THROWL THEN  /*  COLLAPSE KL LIST     */
                |              DO ;
 665    4  1    |                  I = 0 ;
 666    4  1    |                  DO IKL = 1 TO KLC(J) ;
 667    4  2    |             INCRL:    I = I + 1 ;
 668    4  2    |                       IF KL(J,I) = 0 THEN GO TO INCRL ;
 669    4  2    |                       KL(J,IKL) = KL(J,I) ;
 670    4  2    |                  END ;
 671    4  1    |              END ;
 672    4  0    |         RETURN ;
                |
 673    4  0    |     END CHECKE ;
                |
                |/* *************************************************************
                |   ********************************************************** */
                |
 674    3  0    | CKBE:   PROCEDURE(BP) REORDER ;
                |
                |         /*  THIS PROCEDURE CHECKS EARLIER CONDITION 4 INTERVALS
                |         IN G(E) FOR POSSIBLE COVERAGE BY THE INTERVAL REPRESENTED
```

```
STMT LEV NT

                 |        BY THE INTERVAL BP.  ANY PREVIOUS INTERVALS SO COVERED
                 |        ARE THROWN OUT OF G(E) AND ITS ENTRY IS DELETED FROM
                 |        THE KB LIST.                                         */
                 |
 675    4  0     |        DECLARE BP BIT(*) ALIGNED,
                 |             THROW BIT(1) ALIGNED STATIC,
                 |             (I,IKB,P) FIXED BINARY(15) STATIC ;
                 |
 676    4  0     |        THROW = '0'B ;
 677    4  0     |        DO IKB = 1 TO KBC(J) ;
 678    4  1     |             P = KB(J,IKB) ;
 679    4  1     |             IF (BP & G.INT(P)) = G.INT(P) THEN /* INPUT INTERVAL
                 |                            COVERS OLD ONE; THROW OUT OLD ONE.  */
                 |                  DO ;
 680    4  2     |                       NES = NES - 1 ;
 681    4  2     |                       CALL REMOVEGN(P) ;
                 |                       /*  REMOVE POINTER FROM KB LIST     */
 682    4  2     |                       KBC(J) = KBC(J) - 1 ;
 683    4  2     |                       KB(J,IKB) = 0 ;
 684    4  2     |                       THROW = '1'B ;
 685    4  2     |                  END ;
 686    4  1     |        END ;
 687    4  0     |        IF THROW THEN  /*  COLLAPSE KB LIST    */
                 |             DO ;
 688    4  1     |                  I = 0 ;
 689    4  1     |                  DO IKB = 1 TO KBC(J) ;
 690    4  2     |             INCR:     I = I + 1 ;
 591    4  2     |                       IF KB(J,I) = 0 THEN GO TO INCR ;
 692    4  2     |                       KB(J,IKB) = KB(J,I) ;
 693    4  2     |                  END ;
 694    4  1     |             END ;
                 |
 695    4  0     |        RETURN ;
                 |
 696    4  0     |    END CKBE ;
                 |
                 | /* ****************************************************************
                 |    **************************************************************** */
                 |
 697    3  0     |REMOVEGN:     PROCEDURE(P) 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 REMOVEGN 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.                    */
                 |
 698    4  0     |        DECLARE P FIXED BINARY(15) ;
                 |
```

```
STMT LEV NT

 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) ;                              |
 701   4  0 |          IF G.BPTR(P) ¬= 0 THEN                                            |
            |               G.FPTR(G.BPTR(P)) = G.FPTR(P) ;                              |
 702   4  0 |          CALL RETURNG(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) ;                             |
 705   4  0 |          IF P = LGPTR THEN LGPTR = G.BPTR(P) ;                             |
 707   4  0 |          GO TO SETPTRS ;                                                   |
            |                                                                            |
 708   4  0 |     END REMOVEGN ;                                                         |
            |                                                                            |
            |/* ********************************************************************:  |
            |     *************************************************************** */   |
            |                                                                            |
 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 RETURNG(NFGPTR,NLGPTR) ;                           |
 714   4  1 |                      CALL RETURNG(FGPTR,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 ;                                                             |
            |                                                                            |
            |/* *******************************************************************    |
            |     *************************************************************** */   |
            |                                                                            |
```

```
STMT LEV NT

 723   3  0 |    RETURNG: 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.   */
            |
 724   4  0 |         DECLARE (FIRST,LAST) FIXED BINARY(15) ;
            |
 725   4  0 |         IF STRACE THEN PUT SKIP(1) EDIT
            |              ('RETURNG',FIRST,LAST)  (A,(2)F(15)) ;
            |
 726   4  0 |         G.BPTR(FIRST) = LASTFG ;  /*  SET FIRST BACK POINTER TO POINT
            |                                   TO THE LAST FREE G ELEMENT      */
 727   4  0 |         G.FPTR(LAST) = 0 ;   /*  MAKE SURE LAST FORWARD POINTER IS
            |                                  NULL     */
 728   4  0 |         IF LASTFG ¬=0 THEN G.FPTR(LASTFG) = FIRST ;  /*  SET PREVIOUS
            |                                               LAST FORWARD POINTER   */
 729   4  0 |         LASTFG = LAST ;            /*  MOVE UP LASTFG POINTER   */
 730   4  0 |         IF NEXTFG = 0 THEN NEXTFG = FIRST ;
 731   4  0 |    RETURN;
            |
 732   4  0 |    END RETURNG ;
            |
            |/* *********************************************************************
            |   ******************************************************************** */
            |
 733   3  0 |EINCPLX:      PROCEDURE(I1,GPTR) RETURNS(BIT(1)) 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'B; ELSE '0'B
            |         IS RETURNED.                                        */
            |
 734   4  0 |         DECLARE (I1,GPTR) FIXED BINARY(15) ;
            |
 735   4  0 |         RETURN((G.INT(GPTR) & E(K,I1))=E(K,I1));
            |
 736   4  0 |    END EINCPLX ;
            |
            |/* ********************************************************************
            |   ********************************************************************/
            |
 737   3  0 |PRCOVER: PROC(COVR) REORDER;
 738   4  0 |         DECLARE COVR BIT(*) ALIGNED, STRING BIT(NLMAX) VAR,
            |                (I,M,J,IND) FIXED DECIMAL(3),
            |                NUMB CHAR(3) VARYING;
 739   4  0 |         M=0;
            |
 740   4  0 |         PUT SKIP;
```

```
STMT LEV NT

 741   4  0  |        LINE='COMPLEX:  ';
 742   4  0  |        DO I=1 TO NV;
 743   4  1  |              STRING=SUBSTR(COVR,NVP(I),NLEV(I));
 744   4  1  |              IF STRING=SUBSTR(ONES,1,NLEV(I)) THEN GOTO NEXTV;
 745   4  1  |              IF I>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'B);
 749   4  1  |              IF IND=0 THEN GO TO 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,1)='0'B;
 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),
             |                                 INDEP(ISPL),TOTCOV(ISPL))(COL(100),
             |                                 F(3),X(2),F(3),X(2),F(3),X(2),F(3));
 763   4  3  |                        END;
 764   4  2  |              END;
 765   4  1  |        END;
 766   4  0  |        IF LENGTH(LINE) > 10 THEN PUT SKIP EDIT(LINE)(A);
 767   4  0  |        IF M=0 THEN PUT EDIT(NUMCOV(ISPL),NEWCOV(ISPL),
             |                 INDEP(ISPL),TOTCOV(ISPL))(COL(100), F(3),X(2),
             |                 F(3),X(2),F(3),X(2),F(3));
 768   4  0  |END PRCOVER;
             |
             | /*********************************************************************/
 769   3  0  |TRIPLE: PROCEDURE REORDER;
 770   4  0  |    DCL (COMP(MQPTS)) ALIGNED
             |              BIT(LEN) INIT((MQPTS)'0'B);
             |
 771   4  0  |    DCL (I,J,L,TOTO(MQPTS)) FIXED BIN(15);
 772   4  0  |    TOTO=0;
 773   4  0  |    J=1;
 774   4  0  |    DO I=OMQPTR+1 TO MQPTR;
 775   4  1  |        COMP(J)=MQ.INT(I);
 776   4  1  |        J=J+1;
 777   4  1  |        END;
 778   4  0  |    DO I=1 TO MQPTS;   FINOMQ='1'B;
 780   4  1  |        DO J=1 TO MQPTS;
 781   4  2  |           IF I¬=J THEN DO;
             |
```

```
STMT LEV NT

 782   4  3  |              DO L=1 TO NE(K);                                          |
 783   4  4  |                     IF FINOMQ(L) & (COMP(J) & E(K,L))=E(K,L)            |
             |                     THEN DO;                                            |
 784   4  5  |                        TOTO(I)=TOTO(I)+1;                               |
 785   4  5  |                        FINOMQ(L)='0'B;                                  |
 786   4  5  |                     END; END;                                           |
 788   4  3  |              END;  END;                                                 |
 790   4  1  |              INDEP(I)=TOTCOV(MQPTS)-TOTO(I);                            |
 791   4  1  |           END;                                                          |
             |                                                                         |
 792   4  0  |       END TRIPLE;                                                       |
             |   /* ********************************************************************|**
             |      ********************************************************/          |
             |                                                                         |
 793   3  0  |FRECORE:  PROC RETURNS(FIXED BIN(31)) REORDER;                           |
 794   4  0  |DCL (LAST,L,P) POINTER, (I INIT(0), BL BASED(P)) FIXED BIN(31),          |
             |ALLOC(3) FIXED BIN(31) BASED(L), (FO,FIELD(0:38)) POINTER BASED(L),      |
             |NFIELD(0:40) BIT(32) BASED(L), MVTFLAG BIT(1);                           |
 795   4  0  |P=ADDR(L);  /* L AND BL ARE NOW EQUIVALENT. */                           |
 796   4  0  |BL=16;  /* NOW L POINTS TO LOCATION OF THE CVT POINTER. */               |
 797   4  0  |L=FO;  /* NOW L POINTS TO THE CVT. */                                   |
 798   4  0  |MVTFLAG=SUBSTR(NFIELD(29),4,1)='1'B; /* SET FLAG IF AN MVT SYSTEM. */    |
 799   4  0  |L=FO;  /* L NOW POINTS TO TCBWORDS. */                                  |
 800   4  0  |L=FIELD(1);  /* NOW L POINTS TO THE TASK CONTROL BLOCK. */              |
 801   4  0  |IF MVTFLAG THEN DO;                                                     |
 802   4  1  |L=FIELD(38);  /* NOW L POINTS AT THE DUMMY PARTITION QUEUE ELEMENT-8 */ |
 803   4  1  |L=FIELD(2);  /* NOW L POINTS TO THE PARTITION QUEUE ELEMENT */          |
 804   4  1  |LAST=FIELD(1);  /* NOW LAST POINTS TO THE LAST FREE QUEUE ELEMENT */     |
 805   4  1  |DO WHILE(L¬=LAST);  /* ADD UP THROUGH ALL THE QUEUE ELEMENTS */          |
 805   4  2  |L=FO;  /* L POINTS TO NEXT FBQE */                                      |
 807   4  2  |I = I + ALLOC(3);  /* ADD UP THE SIZE OF THIS FREE BLOCK. */            |
 808   4  2  |END;                                                                    |
 809   4  1  |END;                                                                    |

 810   4  0  |ELSE MFT_OR_PCP:  DO;                                                   |
 811   4  1  |L=FIELD(6);  /* L NOW POINTS TO THE BOUNDARY BOX */                     |
 812   4  1  |L=FO;  /* L POINTS TO THE FIRST FREE QUEUE ELEMENT. */                  |
 813   4  1  |DO WHILE(BL¬=0);                                                        |
 814   4  2  |I = I + ALLOC(2);  /* ADD SIZE OF THIS FREE BLOCK. */                   |
 815   4  2  |L=FO;  /* POINT L AT NEXT FREE QUEUE ELEMENT. */                        |
 816   4  2  |END;                                                                    |
 817   4  1  |END;                                                                    |

 818   4  0  |RETURN(I/1024);  /* RETURN ANSWER IN K-BYTE UNITS. */                   |
 819   4  0  |END FRECORE;                                                            |
             |                                                                         |
             |/************************************************************************|
             |************************************************************************/|
             |                                                                         |
```

```
STMT LEV NT

 820   3  0  |  END MAINBLK;                                                        |
 821   1  0  |  FINISH:  /*===========*/   END AQ7;   /*====================*/      |
```

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-75-731	2.	3. Recipient's Accession No.
4. Title and Subtitle AQVAL/1 (AQ7) User's Guide and Program Description			5. Report Date June 1975
			6.
7. Author(s) James Larson and R. S. Michalski			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801			10. Project/Task/Work Unit No.
			11. Contract/Grant No. NSF DCR 74-03514
12. Sponsoring Organization Name and Address National Science Foundation Washington, D.C.			13. Type of Report & Period Covered
			14.

15. Supplementary Notes

16. Abstracts 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).

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

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

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

FORM NTIS-35 (10-70) USCOMM-DC 40329-P71