79-2

Report No. UIUCDCS-R-79-949

The Uniclass Inductive Program AQ7UNI: Program Implementation and User's Guide

by

Robert Stepp

July 1979

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

This work was supported in part by the National Science Foundation under Grant NSF MCS 79-06614.

C O N T E N T S

ABSTRACT

This paper contains implementation notes and user's guide for an inductive program (AQ7UNI) which given a set of events (via an integer-valued feature vector for each object), generates one or more characterizations of those events, expressed in the form of VL_1 expressions. Variable-valued Logic System VL_1 is a monadic predicate calculus in which rules can be formed which describe single events or sets of events. The VL_1 characterization is a generalization of the descriptions of the event examples given to the program. The degree of generalization is controlled by the user.

AQ7UNI belongs to a family of programs which employ quasi-extremal optimality techniques. Data input formats are highly compatible with the discrimination generating program AQVAL/1(AQ7). A variety of operational parameters are provided to direct the generalization processes and to determine the quasi-optimality judging criteria and tolerances.

terms and complexes become equivalent, one making a logical statement, the other a set-theoretical statement, about the same situation. Throughout the remainder of this paper the words "term" and "complex" will be used interchangeably, each connoting the hidden properties of the other.

A cover is a set of complexes (a list of terms) such that every event is in the union of the complexes. If the intersection of any two distinct complexes is non-empty, the complexes are intersecting, otherwise they are disjoint.

The variables in system VL_1 may be of nominal or interval scale. Nominal scale variables may be simple, called FACTOR type variables, or generalization tree structured, called STRUCTURED type variables. Interval scale variables are called INTERVAL type variables. A syntactic limitation is built into the VL_1 selector reference set notation. FACTOR variable reference sets may be any powerset of the domain of the variable. STRUCTURED variable reference sets must be a single leaf or node in the generalization tree. INTERVAL variable reference sets must be a single interval subset of the domain of the variable. Because of term/complex equivalency, these syntactic restrictions also further restrict the subsets of events which are legal complexes.

Background

The uniclass inductive program AQ7UNI accepts a set of symbolic descriptions (events) of arbitrary objects and produces a general description (characterization) of the set, expressed in the language VL_1 (Variable-valued Logic system 1 [Michalski 74, 75]). This report describes the AQ7UNI program and explains how to use it.

The Uniclass Algorithm

The basic uniclass algorithm was developed by R. S. Michalski and was partially implemented by a student at the University of Illinois, H. Yuen. The following discussion will explain the algorithm as it exists in a modified and extended form which is the basis for the implementation of the inductive program AQ7UNI, version 2.

In the VL_1 system [Michalski 74], a cover for a class of events is a logical formula which is the disjunction of logical expressions called terms. It has already been shown, by examples, how a term is the product of selectors, and how a term may be satisfied by one or more events. A complex is a subset of the set of all points in the event space. For every term there is an associated complex composed of all those points in the event space at which the term is satisfied. Some complexes cannot be represented by a single term. Such complexes will be purposely avoided by requiring that any complex by exactly described by some term, and with this constraint

#C as defined in the previous definition of density. When event set E is described by complex C, each event in E is being described by an enclosing subspace containing an average of #C/#E points. This gives "generality" or "uncertainty" to the location of the event (it is one of the #C/#E points, but which one is it?). The degree of generalization is the average amount of information disregarded in determining the location of the event when it is described by C(E) rather than E.

rank R(e,e')

R is a measure of the dissimilarity of the events e and e'. Recall that both e and e' are sequences of n values for the n variables. Let d(x,x') by 0 if x=x' and 1 otherwise. Then $R(e,e') = \sum_{i=1}^{n} d(x_i,x_i')$ where x_i $1 \leq i \leq n$ is the sequence of values for e and x_i' $1 \leq i \leq n$ is the corresponding sequence for e'. Rank R is a special case of a more complete dissimilarity measure for VL_1 events found in [Michalski & Larson 78] which applies to nominal, interval, and structured variables. The dissimilarity measure used in this report corresponds to the general treatment the authors cited give to nominal scale variables.

The general uniclass algorithm is diagrammed in figure 1. The symbolic notation used is defined below.

E_L - the set of events

E_R - the set of events remaining to be covered ($E_R \subseteq E_L$)

K - a user-specified number of neighborhoods to build in parallel and from which a "best neighborhood" is selected

E_N^i - the set of events belonging to the ith neighborhood ($1 \leq i \leq K$)

$SEED_i$ - an event selected at random from set E_R which becomes the nucleus of neighborhood i (no two neighborhoods may have the same SEED)

COVER - a set of complexes of the best neighborhoods

Some definitions are needed to assist with a formal presentation of the uniclass algorithm.

variable X_i

X_i $1 \leq i \leq n$, is an integer-valued variable representing some feature of an event.

event e

$e \equiv \{x_1, x_2, \ldots, x_n\}$ is a sequence of n values for the n corresponding variables

event set E

$E \equiv \{e_1, e_2, \ldots, e_m\}$ is a set of m events

selector reference set r_i

$r_i \equiv \{v_{i1}, v_{i2}, \ldots, v_{is_i}\}$ $1 \leq i \leq n$, is a set of s_i values in the domain of variable X_i

selector $[S_i]$

$[S_i] \equiv [X_i = r_i]$ is a logical expression in the VL_1 system which is true only when the value of variable X_i is an element of the set r_i

complex C(E)

$C(E) \equiv [S_{k_1}][S_{k_2}]\ldots[S_{k_j}]$ $1 \leq j \leq n$ is a conjunction of selectors which is true for all events in set E and false for the maximum number of events not in set E. C(E) is both a term in a VL_1 logical expression and the corresponding subspace of the event space.

density of complex (D(C(E))

$D(C(E)) \equiv \frac{\#E}{\#C}$ is the ratio of the number of events in set E to the number of points in the event subspace C(E).

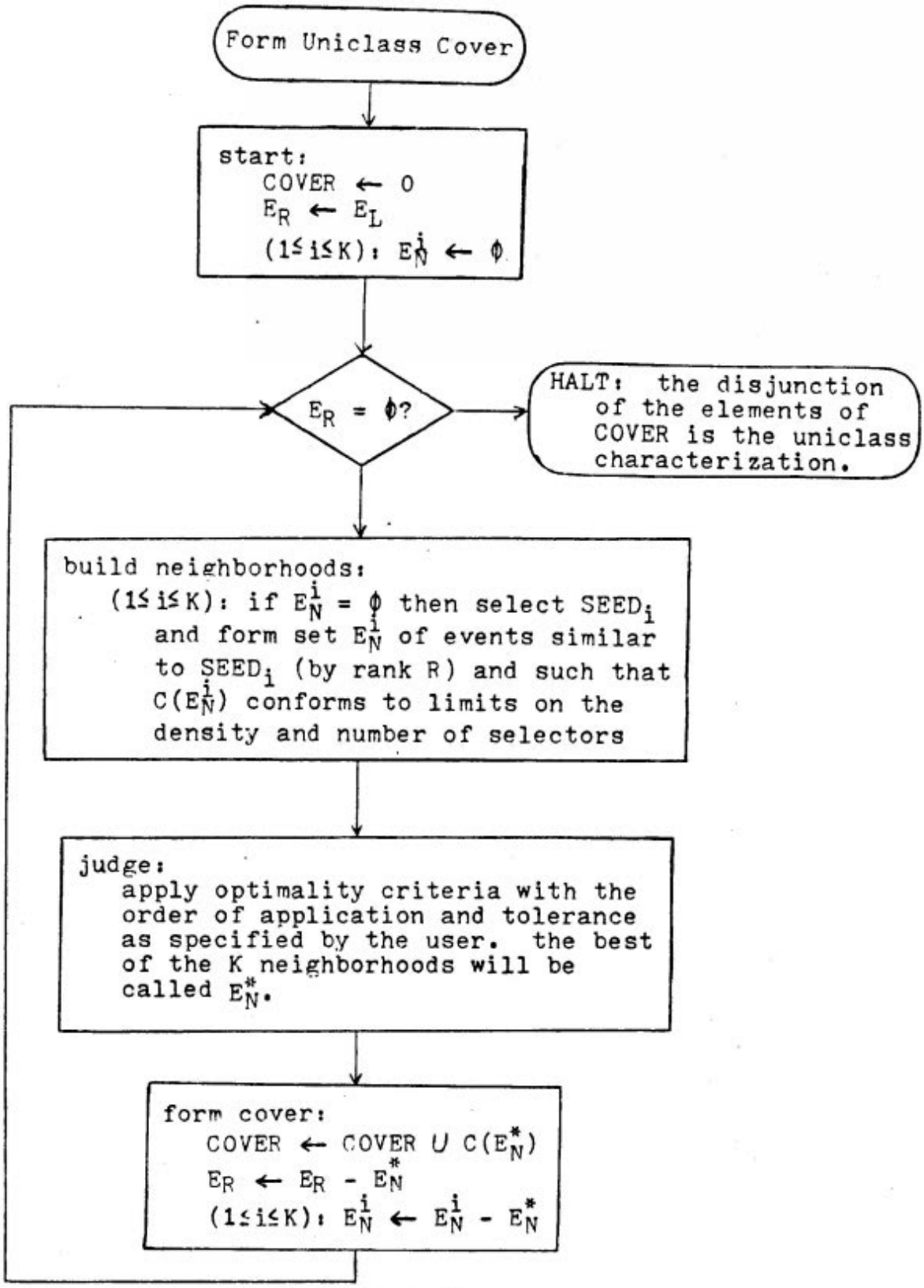
degree of generalization AG(E)

$AG(E) \equiv -\log_2 D(C(E))$ (introduced in [Michalski 79]) is the average number of bits of information needed to locate a particular event from E within a unit event subspace of size #C/#E, with #E and

The uniclass procedure begins when a set of events $E_R = E_L$ has been established from user input data. The following steps are repeated until $E_R = \phi$ and on each iteration another complex in the cover is produced, and the events which it covers are removed from E_R.

step 1: Build neighborhoods. A number (given by the user) of "neighborhoods" are constructed. A neighborhood is a set of events E_N such that for each event e' in E_N, R(e',seed) is not greater than a limit rank set by the user. "Seed" is an event selected arbitrarily from the set E_R and is unique to each neighborhood constructed. $C(E_N)$ is the complex which covers the events E_N. The degree of generalization of $C(E_N)$ is determined by the set E_N. During the neighborhood construction process some events are either excluded from E_N or forced into E_N in order to satisfy the user-given constraints of density threshold and/or selector threshold.

step 2: Select best neighborhood. A quasi-optimal neighborhood is selected according to one or more neighborhood judging criteria (ties are broken by making an arbitrary choice). Seven criteria are defined as follows.



Uniclass Algorithm
Figure 1

AQ7UNI Program Implementation Notes

AQ7UNI is a 1280 statement PL/I program which will run in 140K bytes of memory. The source listing for AQ7UNI has been reproduced in Appendix I and you are directed to that listing for detailed information. The following block diagram shows the procedures into which the program is divided. Each block in the diagram will be discussed briefly.

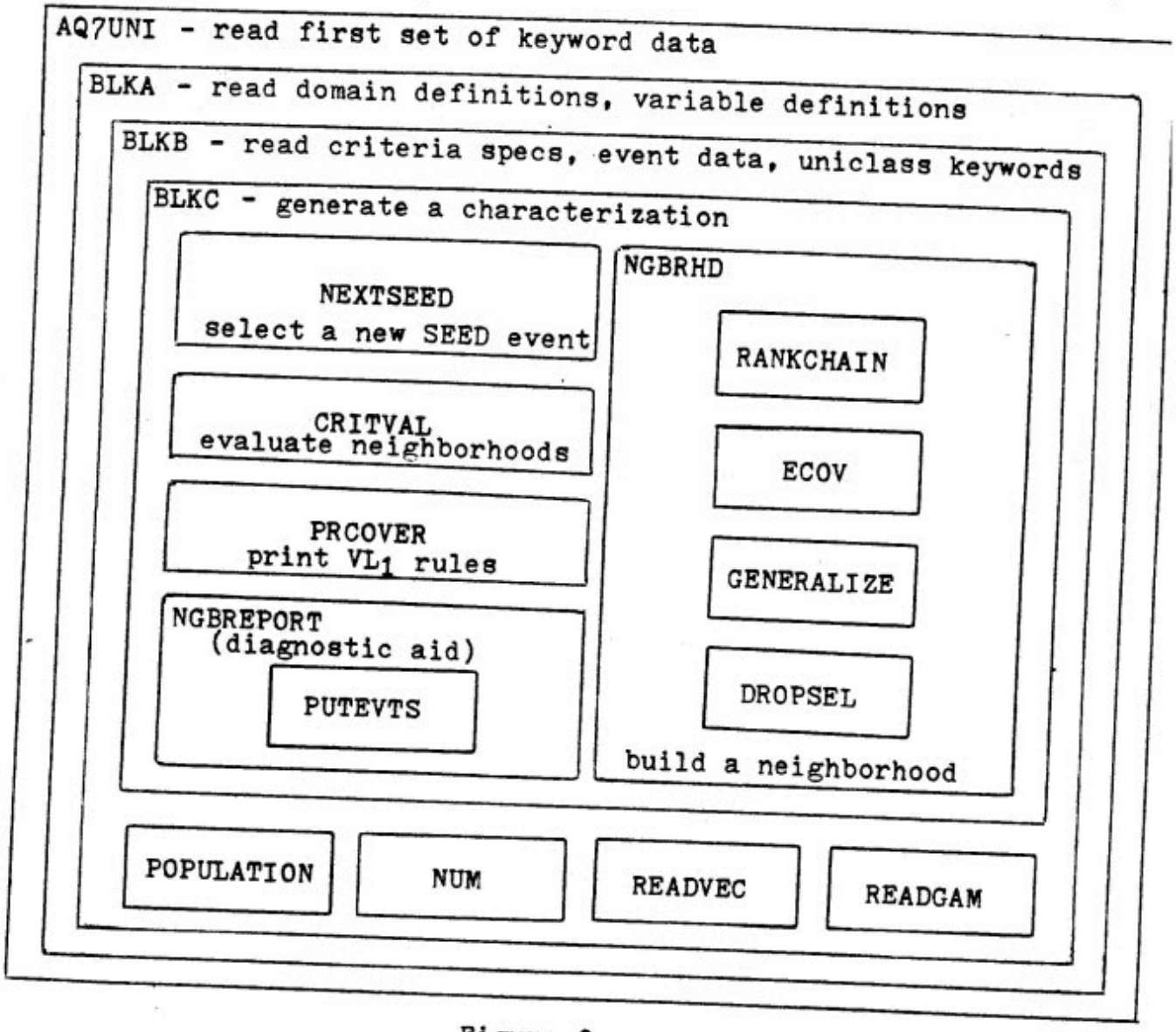


Figure 2

criterion 1: the number of complexes in the cover (estimated as the negative of the number of events in E_R covered by the complex)

criterion 2: the number of selectors in a complex

criterion 3: the sum of the costs of the variables in a formula (costs supplied by the user)

criterion 4: the degree of generalization (estimated as $1/D(C(E_N))$)

criterion 5: the sum of weights of the events covered by the complex (weights supplied by the user)

criterion 6: the length of references ($\sum_i s_i$)

criterion 7: the relative scope of references (the sum of the mean deviations for all variables)

These criteria are identical to the criteria available in program AQ7 [Larson & Michalski 75].

Neighborhood judging may be based on several criteria, applied in an ordering determined by the user. A tolerance value is specified for each and at each step of the judging, a neighborhood is eliminated if its criteria value is greater than an upper bound calculated as

UBOUND = MIN + TOLERANCE * (MAX-MIN).

step 3: Processing the chosen neighborhood. The best neighborhood represents a complex $C(E_N)$ which will be saved to become one complex in the cover of the events. The events in E_N are eliminated from E_R and from other neighborhoods which were not selected.

As long as the bit string is not all zero, neighborhoods are constructed using the NGBRHD procedure and the CRITVAL procedure then determines which one is quasi-optimal according to the criteria requested. As each neighborhood is being built up, a record is kept (as a bit string) of each event which the complex of the neighborhood, $C(E_N)$ covers. After CRITVAL has determined which neighborhood is the best, the set E_R is updated via a simple bit string operation to remove those bits corresponding to covered events, i.e. events in E_N. The POPULATION procedure can count the bits in a bit string whenever a population count is necessary. When E_R becomes empty, the procedure PRCOVER is used to convert the internal representation of the complex into standard notation and print the results.

NEXTSEED

The NEXTSEED procedure selects an arbitrary event from E_R for use as the seed event in neighborhood construction.

CRITVAL

CRITVAL applies whatever criteria the user requests to all existing neighborhoods and then makes a selection of the neighborhood which is optimal according to the criteria used.

PRCOVER

PRCOVER prints out the complexes given a bit string which is the internal form for both complexes and events. PRCOVER also has access to the variable and value names

AQ7UNI

The main program block is responsible for beginning to set up the environmental data needed later. The first set of keyword data (NGE,NMQ,LQTRACE,STRACE,QLQT,QST,PNTE, MODE,SAVE,SAVELQ,MAXSTAR, CUTSTAR,INFORM,TITLE,LQST,CLASSES, MAXNV,DOMAINS,NAMES,MAXNAMELEN,STLVLS) is read by the main block. Each problem starts out here and then control flows to BLKA, BLKB, and BLKC as more parameters are read and evaluated.

BLKA

BLKA is responsible for reading domain definitions, variable definitions, the number of levels for each variable and the number of classes and the number of sample events in each.

BLKB

BLKB takes over and reads the ordering information, the criteria specification, the event data, and the uniclass keyword data (UNICLASS,QUT,UNITRACE,SAVEC). Then BLKB reads in each characterization specification (the number of such specifications is given by the UNICLASS keyword parameter). Block BLKC is invoked for each characterization.

BLKC

Most of the work is done here and BLKC is built much as the flow chart on page 8 indicates. Using the ULIST variable, the event set E_R is built from the events in the indicated classes. E_R is actually a bit string in the program with each event represented by a unique bit.

NGBRHD

NGBRHD builds neighborhoods around a given seed event. The algorithm for neighborhood construction is given in [Stepp 79] and in figure 3. The procedure RANKCHAIN creates RANK+1 singly linked lists of events of E_R according to their rank with respect to the seed event. NGBRHD then goes through a trial and error procedure of adding events to the neighborhood, first an entire chain of events of the same rank at a time, but later, event by event until it can go no further. The last legal neighborhood under the constraints of rank, disjointness (optional), density threshold, and selector threshold is the final result. Along the way, the GENERALIZE procedure is used to edit the values which interval or structure variables may take. If the selector threshold is not satisfied by the final arrangement, the DROPSEL procedure is used to further reduce the number of selectors. NGBRHD reports the density of its result and gives bit strings which indicate which events the neighborhood covers.

ECOV

ECOV checks lists of events to determine which ones are covered by the complex for a neighborhood. The test involves only simple bit string operations.

tables. Numeric variable values and event data values are used throughout the program from input up to the time to print the complex. User supplied names are substituted for numeric values as the complexes are printed out.

NGBREPORT

The NGBREPORT procedure is used only when the "quick trace" is active (the QUT parameter). NGBREPORT prints data associated with each neighborhood on each cycle of the program. The internal procedure PUTEVTS is used to print lists of events which are included in the neighborhood.

POPULATION

POPULATION counts one bits in a bit string. Since extensive use is made of bit strings in AQ7UNI, an efficient population count procedure is important.

NUM

NUM is a utility procedure which can convert a number to a varying length character string, or to the correct special name associated with the given value. NUM is used most by the PRCOVER procedure.

READVEC

READVEC reads event data which is in vector format.

READGAM

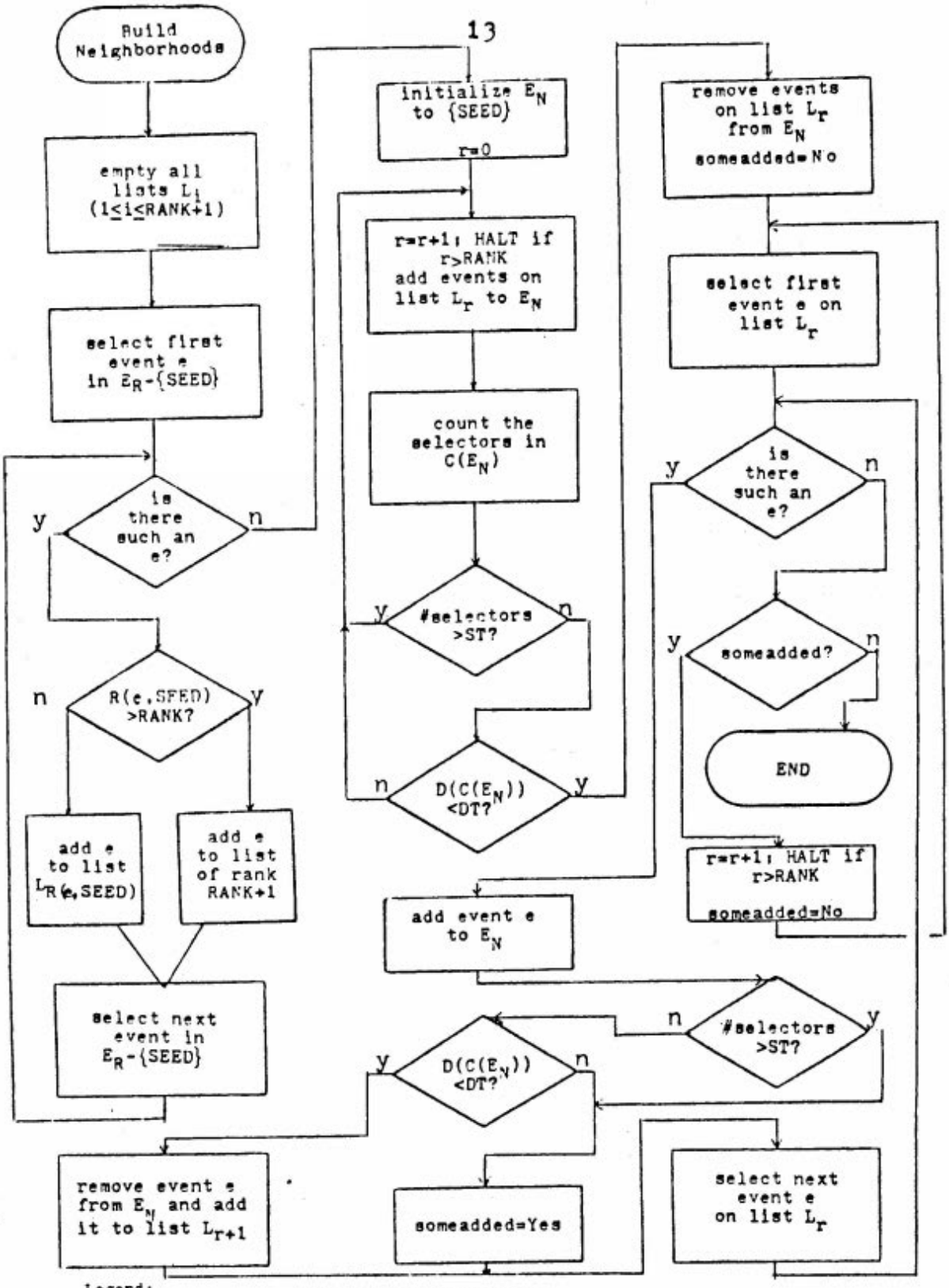
READGAM reads event data which is in gamma format. Gamma format is not often used and is described fully in [Larson 75].

GENERALIZE

GENERALIZE extends the reference set for variables of interval or structure type. Interval variables are extended by filling in gaps of missing values in an enclosing interval, because in system VL_1, the reference set for a variable of interval type must be a single closed interval. Structure variables are extended by finding a common node in the variable structure which encompasses all values present. After GENERALIZE, every interval variable has a reference set which represents only one interval and every structure variable has a reference set which represents a particular node in the structure tree.

DROPSEL

The selector threshold parameter is used to cause all but a certain number of selectors to be extended and dropped. Sometimes the neighborhood construction process is not sufficient to obtain the desired number of selectors. (That is not surprising since the neighborhood construction process is not involved with the number of selectors.) When the number of selectors remains too large, DROPSEL finds and extends references on certain selectors so that they are eliminated. First, variables of factor type are considered. DROPSEL eliminates those with the highest fraction of their domain in the reference set. If that does not reduce the



Legend:
- SEED: an event which was selected from E_R
- E_R: the set of events remaining to be covered
- L_i: a list of events whose distance to SEED (measured by R) is i
- RANK: a rank limit imposed by the user
- E_N: the set of events belonging to the neighborhood
- $C(E_N)$: the smallest single complex covering the events E_N
- ST: the maximum-number-of-selectors threshold
- DT: the minimum-density threshold
- D(C): the density of complex C

Neighborhood Construction
Figure 3

User's Guide for Uniclass Program AQ7UNI

(This user's guide is for version 2 of
AQ7UNI - November 1978.)

Robert Stepp

selector count sufficiently, then all variables are considered and again those with the highest fraction of the domain already present in the reference set are eliminated. When AQ7UNI is producing disjoint complexes, care is taken not to eliminate any variable needed to maintain disjointness. When disjoint complexes are being generated, DROPSEL may not always be able to reduce the number of selectors to the limit given by the selector threshold.

A note About the Internal Bit Representation of Events

Individual events and complexes are represented within AQ7UNI as bit strings. The strings are all the same length, with a bit for each value of each variable. This works because AQ7UNI permits only integer values for its variables and the underlying numerical domains of all variables begin with zero. In actual practice, more than the minimum number of bits are consumed, so as to be able to align the 0-bit for each variable at a character (byte) boundary. Whenever possible, the bit strings are operated on as character strings to improve program efficiency.

MODE=

default: MODE='IC'

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

MODE is an AQVAL/1 parameter not used by AQ7UNI.

INFORM=

default: INFORM='VECTOR'

possible values: 'VECTOR', 'GAMMA'.

INFORM determines the format of the event description.

TITLE=

default: TITLE=0

possible values: any non-negative integer.

TITLE specifies the number of lines of title data.

MAXSTAR=

default: MAXSTAR=150

possible values: any positive integer less than NGE.

MAXSTAR is an AQVAL/1 parameter not used by AQ7UNI.

CUTSTAR=

default: CUTSTAR=50

possible values: any positive integer less than NGE.

CUTSTAR is an AQVAL/1 parameter not used by AQ7UNI.

NGE=

default: NGE=200

possible values: any positive integer.

NGE is an AQVAL/1 parameter not used by AQ7UNI.

NMQ=

default: NMQ=35

possible values: any positive integer.

NMQ defines the storage area for the final output complexes generated by AQ7UNI.

Introduction

AQ7UNI Version 2 is a PL/I program which can accept one or more descriptions of classes of events and generate one or more characterizations for each. The input formats are purposefully similar to those used by AQVAL/1 (AQ7) [Larson-75]. Only a few statements of uniclass parameters need be appended to data already set up for AQVAL/1 (AQ7) to use AQ7UNI to generate characterizations of the events. Version 2 of AQ7UNI supports the following parameters and features not available in previous versions.

- interval variable domains
- structured variable domains
- user selected judging criteria
- domain definitions with symbolic assignments to variables and their values

Input Parameters

Input to AQ7UNI is defined below. Many input specifications match those of AQVAL/1 (AQ7), though some of the parameters for AQVAL/1 (AQ7) are accepted by AQ7UNI but cause no action. The parameters for AQ7UNI are either simply numbers separated from each other by blanks, or keyword expressions. Whenever simple numerical values are called for they must be coded. Keyword expressions, however, need not be coded when the default value is desired.

SAVELQ=

default: SAVELQ='0'B

possible values: '0'B, '1'B

SAVELQ is an undocumented AQVAL/1 parameter not used by AQ7UNI

PNTE=

default: PNTE='1'B

possible values: '0'B, '1'B

PNTE controls the printing of the input event sets. PNTE='1'B causes printing to be performed.

CLASSES=

default: CLASSES=32

possible values: any non-negative integer

CLASSES specifies the maximum number of event classes.

MAXNV=

default: MAXNV=32

possible values: any non-negative integer

MAXNV specifies the maximum number of variables.

DOMAINS=

default: DOMAINS=0

possible values: any non-negative integer

DOMAINS specifies the number of domain definitions. When DOMAINS>0 domain definition parameters are required. See the special section on domain definitions.

LQST=

default: LQST='1'B

possible values: '0'B, '1'B.

LQST is an AQVAL/1 parameter not used by AQ7UNI.

SAVE=

default: SAVE='0'B

possible values: '0'B, '1'B.

SAVE is an AQVAL/1 parameter not used by AQ7UNI.

QLQT=

default: QLQT='0'B

possible values: '0'B, '1'B.

QLQT is an AQVAL/1 parameter not used by AQ7UNI.

LQTRACE=

default: LQTRACE='0'B

possible values: '0'B, '1'B.

LQTRACE is an AQVAL/1 parameter not used by AQ7UNI.

STRACE=

default: STRACE='0'B

possible values: '0'B, '1'B.

STRACE is an AQVAL/1 parameter not used by AQ7UNI

QST=

default: QST='0'B

possible values: '0'B, '1'B.

QST is an AQVAL/1 parameter not used by AQ7UNI.

Variable definition data: Note: if DOMAINS= is specified and given a value greater than zero, then see the section on domain definition for parameters to be used in lieu of the following items.

One of the following specifications must appear.

(a) n 'FACTOR' <n numbers separated by spaces>

(b) n 'INTERVAL' <n numbers separated by spaces>

default: no default

example: 2 'FACTOR' 1 4

The n numbers on the right denote variables which are of the type indicated (either interval variables or factor variables). Variables not mentioned are assigned the opposite variable type. Structure variables can be defined only via domain definition (see section on domain definition). None of the editorial symbols "(a)", "(b)", "<", or ">" are actually coded.

number of levels per variable:

n, <n numbers separated by spaces>

default: no default

example: 4, 5 3 2 4

The n numbers on the right indicate the number of levels of each of the n variables respectively starting with variable X_1.

NAMES=

default: NAMES=0

possible values: any non-negative integer

NAMES specifies the maximum number of symbolic names for data values which may be defined. (See the section on domain definitions).

MAXNAMELEN=

default: MAXNAMELEN=8

possible values: any non-negative integer

MAXNAMELEN specifies the maximum length of any name in characters. Names longer than this will be truncated.

STLVLS=

default: STLVLS=0

possible values: any non-negative integer

STLVLS specifies the maximum number of structured variable levels. (See the section on domain definitions).

This ends the first group of keyword parameters. A semi-colon must be typed to separate these keywords from the following data items.

Title data: As many lines of title data as were specified via the TITLE= parameter should be included at this point in the input stream.

event data:

At this point in the input stream place a vector of numbers (if INFORM='VECTOR') or a single number (if INFORM='GAMMA') to describe each event. Events must be grouped by class and appear in class order.

Z and/or W values:

Z values are the costs for each variable. W values are the costs for each event. These values are to be coded only if criterion numbers 3 and/or 5 are used as criteria specifications. If criterion 3 is used, then any number of specifications of the following form may be given. After the last one code a semi-colon.

Z(i)=

default: Z(i)=1.0

Z(i) specifies the cost for variable i

If criterion 5 is used, then any number of specifications of the following form may be given. After the last one code a semi-colon.

W(c,n)=

default: W(c,n)=1.0

W(c,n) specifies the cost for the nth event defined for class c.

If both criterion 3 and criterion 5 are specified, then code a set of Z specifications and a set of W specifications with the Z's first if criterion 3 occurs first in the criterion list and with the

number of events per event class:

n, <n numbers separated by spaces>

default: no default

example : 3, 5 4 10

The n numbers on the right indicate the number of events in each event class. These events are defined later on in the input stream.

ordering information:

integers separated by spaces

Ordering information consists of a permutation of the sequence of integers 0,1,... up to one less than the number of classes.

criteria specification:

n, <n integers (1-7)>, <n real numbers (0.0-1.0)>

Up to seven user criteria may be specified. The criteria are defined in the body of this report. The integers serve to identify which criterion is to be used while the real numbers give the tolerance values.

example: 2, 4 6, 0.0 0.5

The example indicates two criteria will be used. Criterion #4 will be applied first with a tolerance of 0.0. Then criterion #6 will be applied with a tolerance of 0.5.

SAVEC='0'B.

A semi-colon must be coded after these keyword parameters.

The following group of specifications occur as many times as the value of the UNICLASS parameter above. Each group of specifications pertains to one characterization performed.

MODE=

default: MODE='REL'

possible values: 'REL', 'EXACT', 'APPROX', 'FREE'

MODE controls the density threshold value. With MODE='EXACT' the density threshold is set to 1.0. With MODE='FREE' the density threshold is 0.0. With MODE='APPROX' the density threshold is given by the DT= parameter and with MODE='REL' the density threshold is the product of the value given by the DT= parameter and the density of the learning sample in the event space.

DT=

default: DT=0.0

possible values: any non-negative real number

The DT value is used only when MODE is 'APPROX' or 'REL'.

ST=

default: ST=number of variables

possible values: any positive integer

W's first if criterion 5 occurs first in the list. This concludes the parameter specifications which AQ7UNI and AQVAL/1 (AQ7) have in common. The following parameters are peculiar to AQ7UNI.

uniclass parameters:

UNICLASS=

default: UNICLASS=0

possible values: any non-negative integer

UNICLASS specifies the number of uniclass characterizations to be performed on the event data.

QUT=

default: QUT='0'B

possible values: '0'B, '1'B

QUT controls an informative "quick trace" of the neighborhood construction process.

UNITRACE=

default: UNITRACE='0'B

possible values: '0'B, '1'B

UNITRACE controls the printing of detailed internal data.

SAVEC=

default: SAVEC='0'B

possible values: '0'B, '1'B

SAVEC determines whether characterization complexes are to be written in internal form to a file with DDname COVER. No such output is produced when

ULIST=

default: a bit string of all ones

possible values: any bit string of the form 'bbbb'B where each b is either 0 or 1. The number of bits must match the number of classes defined. Starting with class 0 on the left, each bit corresponds to one class of events. If the corresponding bit is a 1, then the events of that class will be included in the event set to be characterized. AQ7UNI makes no notice of the class of events, this is merely a technique to facilitate taking certain subsets of the learning sample.

example: (assume two classes) ULIST='01'B

This indicates that class 1 (the second class) of events will be covered. ULIST='11'B would indicate that both classes 0 and 1 are to comprise the events characterized.

A semi-colon must be coded at this point.

criteria choices:

If NCRIT= specifies a value greater than 0, criteria choices must be coded. If NCRIT=n, then n integers in the range 1 to 7 are coded followed by n real numbers. The integers serve to identify which criterion is to be used while the real numbers give the tolerance values. Criteria choices for a given

The ST parameter gives the maximum number of selectors any single complex is to contain.

RANK=

default: RANK=number of variables

possible values: any non-negative integer

RANK limits the degree of dissimilarity between events which make up neighborhoods.

NGB=

default: NGB=3

possible values: any integer greater than 3

NGB specifies the number of neighborhoods built in parallel and from which the quasi-optimal one is chosen. Characterizations should improve as NGB approaches the number of events in the learning sample, but process time grows in proportion to NGB as well.

TYPE=

default: TYPE='IC'

possible values: 'IC', 'DC'

TYPE determines whether intersecting ('IC') or disjoint ('DC') complexes are to be produced.

NCRIT=

default: NCRIT=0

possible values: an integer from 0 to 7

NCRIT specifies the number of user selected criteria choices to be made.

Sample AQ7UNI Input Stream

```
 1   INFORM='VECTOR'
 2   TITLE=1;
 3   THIS WILL BE THE TITLE OF THIS EXAMPLE
 4   4 'FACTOR' 1 2 3 4
 5   4, 3 3 4 3
 6   2, 4 4
 7   0 1
 8   1 1 0.0
 9   2 0 3 0
10   2 2 3 1
11   2 2 1 2
12   2 2 0 1
13   2 2 2 0
14   1 2 3 2
15   2 2 2 1
16   1 2 1 1
17   UNICLASS=2  QUT='1'B;
18   MODE='EXACT'  NGB=5  ULIST='10'  NCRIT=2;
19       4 7 0.25 0.6
20   MODE='APPROX'  DT=.375  RANK=2  ULIST='11'B;
21   *
22   INFORM='VECTOR'
     .
     .
     .
```

line 6: number of classes, number of events per class

line 7: ordering information (the first class will be called class 0, the second, class 1)

line 8: general criteria (criterion #1, tolerance 0.0)

lines 9-16: event data in vector format (first 4 are in class 0, second 4 in class 1)

line 17: Uniclass keywords

lines 18-20: parameters for two characterizations

line 21: problem data separator, followed by next problem in line 22

characterization override the general criteria stated previously (following the ordering information) which serve as the default criteria.

Note: The reading of Z and/or W data values is under the control of the general criteria specification. If you wish to use criterion 3 or 5 you must be sure to also state these same criteria as general criteria to trigger the solicitation of associated data values. The same Z and/or W values are used for all characterizations.

This concludes the parameters for each uniclass characterization. At this point you may place a separator card which is an asterisk in column 1 followed by blanks and then begin coding parameters for another entire problem. You may place as many independent problems (separated by separator cards) in the input stream to AQ7UNI as you wish.

The example on the following page illustrates an entire input stream. Line numbers are for reference only.

lines 1,2: AQVAL/1 keywords

line 3: title data

line 4: factor/interval specification (variables 1,2,3,4 are of factor type

line 5: number of variables, number of levels per variable

you must code that number of names, each name must be enclosed in apostrophes. "Type" indicates the type of the domain and is either FACTOR, INTERVAL, or STRUCTURE. ":dname" is optional and specifies any name you wish to call the domain. The parameter MAXNAMELEN= described early in the user's guide specifies the number of characters stored for each name (or dname). The following example illustrates domain definition. Assume DOMAINS=2 was coded.

```
1  'FACTOR' 4 0
2  'INTERVAL:COLOR' 5 5 'WHITE' 'LITE GRAY'
              'GRAY'  'DARK GRAY'  'BLACK'
```

These two statements define domains #1 and #2. The first statement indicates that domain 1 is of factor type and has four levels and no special names. The second statement indicates that domain 2 is of interval type called "color" with 5 levels and 5 special names. The value 0 will be called "white", the value 1 will be called "lite gray",..., the value 4 will be called "black". The parameter NAMES= must be given a value large enough to accomodate all of the names given to all domain definitions. NAMES= should be the sum of the number of levels (l#) of all domain definitions which have a non-zero number of names (n#).

Domain Definitions

The parameters described on the preceeding pages support those features of AQ7UNI which are compatible with AQVAL/1 (AQ7). AQ7UNI has a domain definition feature which is not compatible with AQVAL/1 (AQ7) and for that reason it is documented separately. The following parameters are to be used whenever the DOMAINS= specification is given a positive value. Parameters through and including the title data are coded as before. None of the variable definition parameters which follow the title data up to (but excluding) the ordering information are to be used when DOMAINS has a positive value. Substitute parameters given below are used instead.

domain definition data:

d# 'type:dname' l# n# <n# number of names>

default: no default

Domains are numbered 1 through DOMAINS (DOMAINS is the value given to the DOMAINS= keyword). Domains are defined in order, each definition being of the form given above. "d#" is the domain number. "l#" is the number of levels associated with this domain. (The domain is the integers 0 through l#-1). "n#" is the number of special names to be assigned to the values in the domain. Following the n# number

```
3 'STRUCTURE:SHAPE' 7 7 'LINE' 'TRIANG' 'CIRCLE'
                        'ELLIPSE' 'HEXAGON'
                        'SQUARE' 'TRAP.'
                    4 4 'CLOSED LINE' '4-SIDED'
                        'POLYGON' 'HAVE AREA'
                    2 2 3
                    2 5 6
                    3 8 1 4
                    2 7 9
```

The domain is named "shape" and has 7 levels which are named as this table indicates: Four internal nodes are to be defined and will be given internal numbers as the extension to the table indicates: (Note that these extended levels may not be found in the input data defining the sample events.)

0	line
1	triang
2	circle
3	ellipse
4	hexagon
5	square
6	trap.
7	closed line
8	4-sided
9	polygon
10	have area

The relationships between the internal nodes and the leaves are given by the four specifications following the last internal node name. "2 2 3" applies to name 7 (the first internal node) and says there are 2 conceptual refinements of "closed line", namely items 2 (circle) and 3 (ellipse). Similarly there are two refinements for "4-sided" which are items 5 (square) and 6 (trap.). "Polygon" has 3 refinements which are 8 (4-sided), 1 (triang) and 4 (hexagon). Lastly, "have area" has 2 conceptual refinements: 7 (closed line) and 9 (polygon).

Domain definition for a structured domain is more complicated. The regular domain defining parameters are coded and then following them use these extra specifications:

sl# sn# <sn# number of names>

nn <nn number of numbers separated by spaces>

A structured domain is a tree. The leaves are defined via the regular portion of the domain definition statement, i.e. the number of leaves is the number of levels (l#). The number of interior nodes is specified by the sl# parameter and via the sn# parameter, each interior node may have a name. Following the list of interior node names (if any) come sl# number of structuring specifications. The structuring specifications correspond to the interior nodes, and give a list of previously defined nodes (either leaves or internal) which are to be considered conceptual refinements.
An example should clear up the mysteries. Consider the set of the objects: line, circle, ellipse, square, trapezoid, triangle, hexagon. Circles and ellipses are closed lines. Squares and trapezoids are 4-sided. 4-sided, triangle, and hexagon objects are polygons. Polygons and closed lines have area. Suppose this kind of domain is to be domain number 3. Then we write:

Sample AQ7UNI Input Stream
using Domain Definitions

```
 1   INFORM=VECTOR  NAMES=3
 2   DOMAINS=2  TITLE=1;
 3   THIS WILL BE THE TITLE OF THIS EXAMPLE
 4   1  'FACTOR' 4 0
 5   2  'FACTOR:SIZE' 3 3 'SMALL' 'MED' 'LARGE'
 6   4, 2 2 1 2
 7   4  '. OF #1'  '. OF #2'  'TYPE'  '. OF BOX'
 8   2, 4 4
 9   0 1
10   1 1 0.0
11   2 0 3 0
12   2 2 3 1
13   2 2 1 2
14   2 2 0 1
15   2 2 2 0
16   1 2 3 2
17   2 2 2 1
18   1 2 1 1
19   UNICLASS=2  QUT='1'B;
20   MODE='EXACT'  NGB=5  ULIST='10'  NCRIT=2;
21      4  7  0.25  0.6
22   MODE='APPROX'  DT=.375  RANK=2  ULIST='11'B;
23   *
     .
     .
     .
```

line 1: the total number of names is 3 (domain and variable names do not count).

line 2: 2 domains will be defined.

line 4: domain #1 is of factor type with 4 levels. No special names are used.

line 5: domain #2 is of factor type with 3 levels given special names. The domain is named "size."

line 6: there are 4 variables. Variables 1, 2, and 4 are defined on domain #2 while variable 3 is defined on domain #1.

line 7: The variables are named as follows: "size of #1", "size of #2", "type", and "size of box".

number of variables and domain number for each

n, <n numbers separated by spaces>

default: no default

This specification follows the last domain definition. n specifies the number of variables and each of the n numbers which follow must be the number of a domain which was previously defined. Many variables may have the same domain.

number of variable names and name for each

m, <m names each enclosed in apostrophes>

default: no default

m may be zero in which case no names may be given. In that case the variables are automatically named Xi where i ranges from 1 to the number of variables. If m is less than the number of variables, then those variables not given special names will retain their Xi form of name. If a given name begins with a period, the period is removed and replaced by the name given to the domain. Thus if a variable is of domain number 2 (as defined in a previous example) and its special name is '.#1' then it will be refered to by the name "COLOR#1" on all output. The following example of a whole AQ7UNI input stream illustrates the variable naming feature.

```
STMT LEV NT                                   AQ7UNI - VERSION 2

   1       0  AQ7UNI: PROC OPTIONS(MAIN);  /*+++ AQ7 WITH UNICLASS +++*/
   2   1   0  DCL (LQTRACE,STRACE,QST,QLQT,LQST,SAVE,SAVELQ) /*AQVAL (AQ7) ONLY*/
                BIT(1) ALIGNED,
              (MAXSTAR,CUTSTAR,NGR) /*AQVAL (AQ7) ONLY*/
                FIXED BIN(15),
              MODE /*AQVAL (AQ7) ONLY*/
                CHAR(2) ALIGNED,

              BPC /* NUMBER OF BITS PER CHARACTER */
                FIXED BIN(15),
              CHARBUF /* WORK AREA FOR TITLE */
                CHAR(80) ALIGNED,
              CLASSES /* MAXIMUM NUMBER OF CLASSES */
                FIXED BIN(15),
              COVER /* FILE TO WHICH COMPLEXES MAY BE WRITTEN */
                FILE STREAM OUTPUT EXTERNAL,
              DEBUG /* CONTROL OF DEBUG OUTPUT */
                BIT(1) ALIGNED,
              DOMAINS /* NUMBER OF DOMAINS DEFINED */
                FIXED BIN(15),
              FB15 /* HALF-WORD WORK AREA, OVERLAYS FB */
                FIXED BIN(15),
            1 FB /* STRUCTURE DEFINED TO COVER FB15 */
                BASED(FP),
            2 (FILL1,FC1) /* THE TWO BYTES IN FB15 */
                CHAR(1) ALIGNED,
              FLUSH /* INDICATES WHEN INPUT DATA MUST BE SKIPPED */
                BIT(1) ALIGNED,
              INFORM /* EVENT INPUT FORMAT (GAMMA OR VECTOR) */
                CHAR(6) ALIGNED,
              MAXNAMELEN /* MAX NUMBER OF CHARS IN EACH NAME */
                FIXED BIN(15),
              MAXNV /* MAX NUMBER OF VARIABLES */
                FIXED BIN(15),
              NAMES /* NUMBER OF DEFINED NAMES */
                FIXED BIN(15),
              NMQ /* MAX NUMBER OF COMPLEXES IN A COVER */
                FIXED BIN(15),
              PRTE /* INDICATES WHETHER TO PRINT EVENT DATA */
                BIT(1) ALIGNED,
              STLVLS /* MAX NUMBER OF STRUCTURE LEVELS */
                FIXED BIN(15),
              TITLE /* NUMBER OF LINES OF TITLE DATA */
                FIXED BIN(15),
              TRDROP /* TRANSLATE TABLE TO DROP LEFTMOST BIT */
                CHAR(256) ALIGNED,
              TRIDX /* LOOKUP TABLE TO GIVE INDEX OF LEFTMOST BIT*/
                (0:255) FIXED BIN(15),
              TRPOP /* LOOKUP TABLE TO GIVE POPULATION OF BYTE */
                (0:255) FIXED BIN(15),

              (I,J,J1,J2,K) FIXED BIN(31);

   3   1   0  BPC = 8;  /* THERE ARE 8 BITS PER CHARACTER */
   4   1   0  TRIDX(0) = 8;
   5   1   0  J1,J2 = 1;
   6   1   0  DO I = 7 TO 0 BY -1;
   7   1   1    DO J = J1 TO J2;
   8   1   2      TRIDX(J) = I;
   9   1   2    END;
  10   1   1    J1 = J2+1;  J2 = (J1*2)-1;
  12   1   1  END;

  13   1   0  TRPOP(0) = 0;
  14   1   0  J1 = 1;
  15   1   0  DO I = 1 TO 8;
  16   1   1    DO J = 0 TO J1-1;
  17   1   2      TRPOP(J+J1) = 1 + TRPOP(J);
  18   1   2    END;
  19   1   1    J1 = J1 * 2;
  20   1   1  END;

  21   1   0  FP = ADDR(FB15);
  22   1   0  FB15 = 0;
  23   1   0  SUBSTR(TRDROP,1,1) = FC1;
  24   1   0  J1,J2 = 1;
  25   1   0  DO I = 1 TO 8;
  26   1   1    DO J = J1 TO J2;
  27   1   2      FB15 = J-J1;
  28   1   2      SUBSTR(TRDROP,J+1,1) = FC1;
  29   1   2    END;
  30   1   1    J1 = J2+1; J2 = (J1*2)-1;
  32   1   1  END;

  33   1   0  DEBUG = '0'B;
  34   1   0  ON ENDFILE(SYSIN) BEGIN;
  35   2   0    PUT SKIP(3) LIST ('END OF DATA ON SYSIN FILE');
  36   2   0    PUT SKIP(2) LIST ('AQ7UNI ENDS');
  38   2   0    STOP;
               END;

               /* BEGIN HERE TO START NEW PROBLEM
                  */
  39   1   0  NEWDATA: CLASSES=32; MAXNV=32; MAXNAMELEN=8;
  40   1   0  MODE='IC';
  41   1   0  INFORM = 'VECTOR' ;
  45   1   0  TITLE,DOMAINS,NAMES,STLVLS = 0;
  46   1   0  NMQ=35;
  47   1   0  PRTE='1'B;
               FLUSH='0'B;
```

APPENDIX I

Program Listing of AQ7UNI
(Version 2)

November 1978

```
STMT LEV NT                                        AQ70NI - VERSION 2

                        FIXED BIN(31),
                    SI  /* STRUCTURE INDEX IN USE */
                        FIXED BIN(31),
                    STRUC /* STRUCTURE DATA */
                      (MAX(1,STLVLS)) FIXED BIN(15),
                    STRUCVARS /* STRUCTURED VARIABLES */
                      (MAXNV) FIXED BIN(15),
                    TYPE /* VARIABLE TYPE */
                      CHAR(MAXNAMELEN+10) VAR ALIGNED,
                    VARNAME /* NAME OF EACH VARIABLE */
                      (MAXNV) CHAR(MAXNAMELEN) VAR ALIGNED,
                    VARTYPE /* TYPE OF EACH VARIABLE */
                      (MAXNV) CHAR(1) ALIGNED,
                    WLEN /* LENGTH OF BIT WORK AREAS IN BITS */
                      FIXED BIN(31),

                    (I,J,K) FIXED BIN(31);

 61   2   0       #IVARS,#SVARS,#AVARS = 0;
 62   2   0       #NAMES,#STRUC,MAISB=1;
 63   2   0       STRUC = 0;
 64   2   0       IF DOMAINS < 1 THEN DO;
                 /* DETERMINE INTERVALS OR FACTORS */

 65   2   1         GET LIST(NSPEC,TYPE);
 66   1   1         IF SUBSTR(TYPE,1,1)='I' THEN DO;
 67   2   2           ANAME = 'F'; TYPE = 'I';
 69   2   2           END;
 70   2   1         ELSE DO;
 71   2   2           ANAME = 'I'; TYPE = 'F';
 73   2   2           END;
 74   2   1         VARTYPE = ANAME;
 75   2   1         DO I=1 TO NSPEC;
 76   2   2           GET LIST(J);
 77   2   2           VARTYPE(J) = TYPE;
 78   2   2           END;

 79   2   1         /* READ NUMBER OF VARIABLES */
 80   2   1           GET LIST(NV);
 81   2   1           IF NV > MAXNV THEN DO;
 82   2   2             PUT SKIP LIST('VARIABLES EXCEED MAXNV');
 83   2   2             FLUSH = '1'B;
 84   2   2             GO TO BLKA_END;
 85   2   2             END;
 86   2   1           DO I=1 TO NV;
 87   2   2             VARNAME(I) = 'X' || SUB(I,0);
                       END;

 88   2   1           IF NSPEC=0 | NSPEC=NV THEN DO;
 89   2   2             PUT SKIP EDIT('ALL VARIABLES WILL BE COVERED BY ') (A);
 90   2   2             IF VARTYPE(1)='F' THEN PUT EDIT('FACTORS') (A);
 91   2   2             ELSE DO;
 92   2   3               PUT EDIT('INTERVALS') (A);
 93   2   3               #IVARS=NV;
 94   2   3               DO I=1 TO NV;
 95   2   4                 INTVARS(I)=I;
 96   2   4                 END;
 97   2   3               END;
 98   2   2             END;
 99   2   2           ELSE DO;
100   2   2             PUT SKIP LIST('THE FOLLOWING VARIABLES ARE '
                         || 'COVERED BY FACTORS');
101   2   2             J1 = 0;
102   2   2             DO I=1 TO NV;
103   2   3               IF VARTYPE(I)='F' THEN DO;
104   2   4                 IF NV>27 THEN J1 = J1 + 1;
105   2   4                 ELSE J1 = I;
106   2   4                 IF J1 > 27 THEN J1 = 1;
107   2   4                 PUT EDIT(I) (COL(35+(J1*3)),F(3));
108   2   4                 END;
109   2   3               END;
110   2   2             PUT SKIP LIST('THE FOLLOWING VARIABLES ARE '
                         || 'COVERED BY INTERVALS');
111   2   2             J1 = 0;
112   2   2             DO I=1 TO NV;
113   2   3               IF VARTYPE(I)='I' THEN DO;
114   2   4                 IF NV > 27 THEN J1 = J1 + 1;
115   2   4                 ELSE J1 = I;
116   2   4                 IF J1 > 27 THEN J1 = 1;
117   2   4                 PUT EDIT(I) (COL(35+(J1*3)),F(3));
118   2   4                 #IVARS=#IVARS+1;
119   2   4                 INTVARS(#IVARS)=I;
120   2   4                 END;
121   2   3               END;
122   2   2             END;

123   2   1         PUT SKIP(2) EDIT('NUMBER OF VARIABLES =',NV) (A,F(4));

                  /* READ NUMBER OF LEVELS */
124   2   1           GET LIST((NL(I) DO I=1 TO NV));
125   2   1           PUT SKIP EDIT('NUMBER OF LEVELS FOR EACH VARIABLE:') (A);
126   2   1           PUT EDIT((NL(I) DO I=1 TO NV)) (COL(38),(27)F(3));
127   2   1           END;
128   2   0       ELSE DO;

129   2   1         /* READ DOMAIN DEFINITIONS */
130   2   1           PUT SKIP(2) LIST ('DOMAIN DEFINITIONS');
131   2   1           PUT SKIP(2);
132   2   1           DO I = 1 TO DOMAINS;
133   2   2             DOMDATA(I).#LVLS,DOMDATA(I).NAMESIDX=0;
134   2   2             DOMDATA(I).STRUCIDX,NI,SI=0;
135   2   2             DOMDATA(I).PREFIX='';
136   2   2             GET LIST (D#,TYPE);
                        IF D# ^= I THEN DO;
```

```
STMT LEV NT                                   AQ7UNI - VERSION 2

                    /* READ AQ7 KEYWORDS
                       */
  48    1  0        GET DATA(NGE,NEQ,LQTRACE,STRACE,QLOT,QST,PNTE,MODE,
                       SAVE,SAVELQ,MAXSTAR,OUTSTAR,INFORM,TITLE,LOST,
                       CLASSES,MAXNV,DOMAINS,NAMES,MAXNAMELEN,STLVLS,DEBUG);
                    /* PROCESS TITLE
                       */
  49    1  0        IF TITLE>0 THEN DO;
  50    1  1          GET SKIP(1);
  51    1  1          DO I=1 TO TITLE;
  52    1  2            GET EDIT(CHARBUF) (A(80));
  53    1  2            PUT SKIP(1) EDIT (CHARBUF) (COLUMN(20),A);
  54    1  2            END;
  55    1  1          END;
  56    1  0        PUT SKIP(2) LIST('AQ7UNI - VERSION 2 - OCT 1978');
  57    1  0        PUT SKIP(2) EDIT('INPUT FORMAT IS ',INFORM)(A,A);
  58    1  0        IF DEBUG THEN PUT DATA(PNTE,DEBUG,INFORM,MODE,PLOSH,
                       NEQ,DOMAINS,NAMES,MAXNV,MAXNAMELEN,CLASSES,STLVLS,TITLE);

  59    1  0        BLKA: BEGIN;
  60    2  0         DCL #E /* NUMBER OF EVENTS */
                         FIXED BIN(31),
                       #IVARS /* NUMBER OF INTERVAL VARIABLES */
                         FIXED BIN(15),
                       #NAMES /* NEXT INDEX OF A NAME */
                         FIXED BIN(31),
                       #NAVARS /* NUMBER OF VARIABLES WITH NA VALUES */
                         FIXED BIN(15),
                       #STRUC /* NEXT INDEX OF A STRUCTURE */
                         FIXED BIN(31),
                       #SVARS /* NUMBER OF STRUCTURED VARIABLES */
                         FIXED BIN(15),
                       ANAME /* WORK AREA FOR DEFINED NAMES */
                         CHAR(MAXNAMELEN) VAR ALIGNED,
                       B#E /* NUMBER OF EVENTS IN BIT UNITS (MULT OF BPC)*/
                         FIXED BIN(15),
                       BLEN /* LENGTH OF EVENT REPRESENTATION IN BITS */
                         FIXED BIN(31),
                       BO /* BIT OFFSET FOR EACH VARIABLE */
                         (MAXNV) FIXED BIN(15),
                       C#E /* NUMBER OF EVENTS IN CHAR UNITS */
                         FIXED BIN(15),
                       CLEN /* LENGTH OF EVENT REPRESENTATION IN CHARS */
                         FIXED BIN(31),
                       CO /* CHAR OFFSET FOR EACH VARIABLE */
                         (MAXNV) FIXED BIN(15),
                       D# /* DOMAIN NUMBER CURRENTLY IN USE */
                         FIXED BIN(31),
                       DO#  /* DOMAIN NUMBER FOR EACH VARIABLE */
                         (MAXNV) FIXED BIN(15),
                     1 DODATA /* DOMAIN DATA */
                         (MAX(1,DOMAINS)),
                     2 #LVLS /* NUMBER OF LEVELS IN THE DOMAIN */
                         FIXED BIN(15),
                     2 DTYPE /* DOMAIN TYPE (I,S,F) */
                         CHAR(1) ALIGNED,
                     2 NAMESIDX /* INDEX TO FIRST DEFINED NAME */
                         FIXED BIN(15),
                     2 PREFIX /* NAME ASSIGNED TO DOMAIN */
                         CHAR(MAXNAMELEN) VAR ALIGNED,
                     2 STRUCIDX /* INDEX TO FIRST STRUCTURE ELEMENT */
                         FIXED BIN(15),
                       IDNAMES /* DEFINED NAMES FOR VARIABLE VALUES */
                         (MAX(1,NAMESL) CHAR(MAXNAMELEN) VAR ALIGNED,
                       INTVARS /* LIST OF INTERVAL VARIABLES */
                         (MAXNV) FIXED BIN(15),
                       LEAF /* VALUE OF A LEAF IN STRUCTURE TREE */
                         FIXED BIN(31),
                       LIMITAL /* AREA OF THE EVENT SPACE */
                         FLOAT,
                       MAICL /* HIGHEST CLASS NUMBER */
                         FIXED BIN(31),
                       MAXEV /* LARGEST NUMBER OF EVENTS IN A CLASS*/
                         FIXED BIN(31),
                       MAXSB /* LARGEST NUMBER OF STRUCTURE BITS */
                         FIXED BIN(31),
                       MXNAME /* NUMBER OF NAMES ALLOWED FOR A DOMAIN */
                         FIXED BIN(31),
                       NAIND /* INDICATES VARIABLES WITH NA VALUES */
                         (MAXNV) CHAR(1) ALIGNED,
                       NAVARS /* VARIABLES WITH NA VALUES */
                         (MAXNV) FIXED BIN(15),
                       NC /* NUMBER OF CHARACTERS REPRESENTING A VARIABLE*/
                         (MAXNV) FIXED BIN(15),
                       NCL /* NUMBER OF CLASSES */
                         FIXED BIN(31),
                       NCMAX /* LARGEST OF THE NC(I) */
                         FIXED BIN(31),
                       NE /* NUMBER OF EVENTS IN EACH CLASS */
                         (0:CLASSES) FIXED BIN(15),
                       NHL /* NUMBER OF INTERIOR NODES IN A STRUCTURE */
                         FIXED BIN(31),
                       NI /* NAMES INDEX IN USE */
                         FIXED BIN(31),
                       NL /* NUMBER OF LEVELS FOR EACH VARIABLE */
                         (MAXNV) FIXED BIN(15),
                       NLVL /* NUMBER OF LEVELS FOR A DOMAIN */
                         FIXED BIN(31),
                       NNAME /* NUMBER OF NAMES ENTERED */
                         FIXED BIN(31),
                       NV /* NUMBER OF VARIABLES */
                         FIXED BIN(31),
                       NSPEC /* NUMBER OF SPECIFICATIONS ENTERED */
```

```
STMT LEV NT                                   AQ7UNI - VERSION 2

 137   2  3          PUT SKIP LIST ('DOMAIN DEFINITION',I,'EXPECTED BUT',
                       D#,'FOUND');
 138   2  3          FLUSH = '1'B;
 139                  GO TO BLKA_END;
 140                  END;
 141                GET LIST (NLVL,NNAME);
 142                DOMDATA(D#).#LVLS = NLVL;
 143                DOMDATA(D#).DTYPE = TYPE;
 144                J = INDEX(TYPE,':');
 145                IF J>0 THEN DOMDATA(D#).PREFIX=SUBSTR(TYPE,J+1);
 146                IF NNAME > 0 THEN DO;
 147                  NINAME = NLVL;
 148                  IF #NAMES+NINAME-1 <= NAMES THEN DO;
 149                    DOMDATA(D#).NAMESIDX = #NAMES;
 150                    #NAMES = #NAMES + NINAME;
 151                    END;
 152                  ELSE NINAME = 0;
 153                  DO J = 0 TO NNAME-1;
 154                    GET LIST(ANAME);
 155                    IF J < NINAME THEN IDNAMES(NI+J) = ANAME;
 156                    END;
 157                  IF NNAME < NINAME THEN DO J = NNAME TO NINAME-1;
 158                    IDNAMES(NI+J) = SUB(J,0);
 159                    END;
 160                  END;
 161                IF DOMDATA(D#).DTYPE = 'S' THEN DO;
 162                  IF MAXSB < NLVL THEN MAXSB = NLVL;
 163                  GET LIST(NBL,NNAME);
 164                  IF MAXSB < NBL THEN MAXSB = NBL;
 165                  IF NNAME > 0 THEN DO;
 166                    NINAME = NBL;
 167                    IF DOMDATA(D#).NAMESIDX > 0 THEN
 168                      IF #NAMES+NINAME-1 <= NAMES THEN DO;
 169                      NI = #NAMES;
 170                      #NAMES = #NAMES+NINAME;
 171                      END;
 172                    ELSE DO;
 173                      #NAMES = DOMDATA(D#).NAMESIDX;
 174                      DOMDATA(D#).NAMESIDX = 0;
 175                      NINAME = 0;
 176                      END;
 177                    ELSE NINAME = 0;
 178                    DO J = 0 TO NNAME-1;
 179                      GET LIST(ANAME);
 180                      IF J < NINAME THEN IDNAMES(NI+J) = ANAME;
 181                      END;
 182                    IF NNAME < NINAME THEN DO J = NNAME TO NINAME-1;
 183                      IDNAMES(NI+J) = SUB(NLVL+J,0);
 184                      END;
 185                    END;
 186                  DOMDATA(D#).STRUCIDX,SI = #STRUC;
 187                  #STRUC = #STRUC + NLVL + NBL;
 188                  IF #STRUC-1 > STLVLS THEN DO;
 189                    PUT SKIP LIST ('STLVLS TO LOW TO ACCOMODATE '
 190                      ||'STRUCTURED DOMAIN NO.',D#);
 191                    FLUSH = '1'B;
 192                    GO TO BLKA_END;
 193                    END;
 194                  DO J = NLVL TO NLVL+NBL-1;
 195                    GET LIST(NV);
 196                    DO K = 1 TO NV;
 197                      GET LIST(LEAF);
 198                      IF LEAF >= J THEN DO;
 199                        PUT SKIP LIST('STRUCTURE DEFINITION ERROR',
 200                          LEAF,'IN LEVEL',J);
 201                        FLUSH = '1'B;
 202                        GO TO BLKA_END;
 203                        END;
 204                      STRUC(SI+LEAF) = J;
 205                      END;
 206                    END;
 207                  END;

/* PRINT DOMAIN DATA */

 208   2  2          IF DOMDATA(D#).DTYPE = 'S' THEN TYPE = 'STRUCTURE';
                     ELSE IF DOMDATA(D#).DTYPE = 'I' THEN TYPE = 'INTERVAL';
                     ELSE TYPE = 'FACTOR';
                     PUT SKIP EDIT('DOMAIN ',D#,' OF ',TYPE,' TYPE HAS ',
 209   2  2          DOMDATA(D#).#LVLS,' LEVELS') (COL(4),A,F(2),(3)A,F(4),A);
 210   2  2          IF DOMDATA(D#).DTYPE ^= 'S' THEN
 211   2  2            IF DOMDATA(D#).NAMESIDX ^= 0 THEN DO;
 212   2  2            PUT EDIT('VALUES WERE NAMED AS FOLLOWS:') (X(4),A);
 213   2  2            NI = DOMDATA(D#).NAMESIDX;
 214   2  2            DO J = 0 TO DOMDATA(D#).#LVLS-1;
 215   2  2              PUT SKIP EDIT(J,IDNAMES(NI+J)) (COL(12),F(4),COL(22),A);
 216   2  2              END;
 217   2  2            END;
                     ELSE;
                     ELSE DO;
 218   2  2            PUT EDIT('STRUCTURE MAP FOLLOWS') (X(6),A);
 219   2  2            SI = DOMDATA(D#).STRUCIDX;
 220   2  2            NI = DOMDATA(D#).NAMESIDX;
 221   2  2            DO J = 0 TO DOMDATA(D#).#LVLS-1;
 222   2  2              PUT SKIP EDIT(' ') (COL(11),A);
 223   2  2              K = 1; LEAF = J;
 224   2  2              DO WHILE (K>0);
 225   2  2                IF NI>0 THEN PUT EDIT(IDNAMES(NI+LEAF)) (A);
 226   2  2                ELSE PUT EDIT(LEAF) (F(3));
 227   2  2                K,LEAF = STRUC(SI+LEAF);
 228   2  2                IF K>0 THEN PUT EDIT(' -> ') (A);
 229   2  2                END;
 230   2  2              END;
 231   2  3            END;
 232   2  3          END;
```

```
STMT LEV NT                                   AQ70NI - VERSION 2

 233    2  2        PUT SKIP(2);
 234    2  2        END;
                  /* READ NUMBER OF LEVELS */
 235    2  1        GET LIST(NV);
 236    2  1        IF NV > MAXNV THEN DO;
 237    2  2          PUT SKIP LIST('VARIABLES EXCEED MAXNV');
 238    2  2          FLUSH = '1'B;
 239    2  2          GO TO BLKA_END;
 240    2  2          END;
 241    2  1        PUT SKIP(2) EDIT('NUMBER OF VARIABLES =',NV) (A,F(4));
 242    2  1        GET LIST((DOM#(I) DO I=1 TO NV));
 243    2  1        PUT SKIP EDIT('DOMAIN NUMBER FOR EACH VARIABLE:')(A);
 244    2  1        PUT EDIT((DOM#(I) DO I=1 TO NV))(COL(38),(27)F(3));
 245    2  1        DO I = 1 TO NV;
 246    2  2          D# = DOM#(I);
 247    2  2          IF D# > DOMAINS THEN DO;
 248    2  3            PUT SKIP LIST('DOMAIN NUMBER FOR VARIABLE',I,
                          'IS INVALID');
 249    2  3            FLUSH = '1'B;
 250    2  3            GO TO BLKA_END;
 251    2  3            END;
 252    2  2          NL(I) = DOMDATA(D#).#LVLS;
 253    2  2          VARTYPE(I) = DOMDATA(D#).DTYPE;
 254    2  2          IF DOMDATA(D#).DTYPE = 'I' THEN DO;
 255    2  3            #IVARS = #IVARS+1;
 256    2  3            INTVARS(#IVARS) = I;
 257    2  3            END;
 258    2  2          ELSE IF DOMDATA(D#).DTYPE = 'S' THEN DO;
 259    2  3            #SVARS = #SVARS+1;
 260    2  3            STRUCVARS(#SVARS) = I;
 261    2  3            END;
 262    2  2          END;
 263    2  1        PUT SKIP EDIT('NUMBER OF LEVELS FOR EACH VARIABLE:')(A);
 264    2  1        PUT EDIT((NL(I) DO I=1 TO NV))(COL(38),(27)F(3));
 265    2  1        GET LIST(ANAME); /* READ VARIABLE NAMES */
 266    2  1        DO I = 1 TO NNAME;
 267    2  2          GET LIST(ANAME);
 268    2  2          IF I <= NV THEN DO;
 269    2  3            IF SUBSTR(ANAME,1,1)='.' THEN
                          VARNAME(I) = DOMDATA(DOM#(I)).PREFIX || SUBSTR(ANAME,2);
 270    2  3            ELSE VARNAME(I) = ANAME;
 271    2  3            END;
 272    2  2          END;
 273    2  1        IF NNAME < NV THEN DO I=NNAME+1 TO NV;
 274    2  2          VARNAME(I) = 'I' || NUM(I,0);
 275    2  2          END;
 276    2  1        END;

 277    2  0      MCMAX = 0; CLEN = 0;  LINITIAL = 1.0;
 280    2  0      DO I = 1 TO NV;
 281    2  1        LINITIAL = LINITIAL * NL(I);
 282    2  1        CO(I) = CLEN+1;
 283    2  1        BO(I) = (CLEN*BPC) + 1;
 284    2  1        MC(I) = (NL(I) + BPC-1)/BPC;
 285    2  1        CLEN = CLEN + MC(I);
 286    2  1        IF MCMAX < MC(I) THEN MCMAX = MC(I);
 287    2  1        END;
 288    2  0      BLEN = CLEN * BPC;
                  /* READ NUMBER OF CLASSES AND NUMBER EVENTS IN EACH */
 289    2  0      GET LIST(NCL);
 290    2  0      MAXCL=NCL-1;
 291    2  0      IF NCL > CLASSES THEN DO;
 292    2  1        PUT SKIP LIST('CLASSES SPECIFICATION ONLY ALLOWS UP TO',
                      CLASSES,'CLASSES');
 293    2  1        FLUSH = '1'B;
 294    2  1        GO TO BLKA_END;
 295    2  1        END;
 296    2  0      GET LIST((NE(I) DO I=0 TO MAXCL));
 297    2  0      PUT SKIP EDIT('NUMBER OF EVENTS',
                    ' SPECIFIED FOR EACH CLASS:','CLASS   #EVENTS')
 298    2  0        (A,A,SKIP(1),COL(8),A);
 299    2  0      PUT EDIT((I,NE(I) DO I=0 TO MAXCL))(SKIP,(2)F(10));
 300    2  0      #E,MAXEV = 0;
 301    2  0      DO I = 0 TO MAXCL;
 302    2  1        #E = #E + NE(I);
 303    2  1        IF MAXEV < NE(I) THEN MAXEV = NE(I);
 304    2  1        END;
 305    2  0      C#E = (#E+BPC-1) / BPC;
 306    2  0      B#E = C#E * BPC;
 307    2  0      WLEN = MAX(B#E, BLEN, MAXSB);
 308    2  0      IF DEBUG THEN PUT DATA(DOMDATA,IDNAMES,STRUC,VARTYPE,DOM#,
                    #NAMES,#STRUC,MAXCL,MAXEV,NCL,INTVARS,#IVARS,STRUCVARS,
                    #SVARS,TYPE,NL,BO,CO,MC,NE,MAXSB,#E,C#E,LINITIAL,BLEN,CLEN,
                    WLEN,MCMAX,VARNAME);

 309    2  0      BLKB: BEGIN;
 310    3  0        DCL BV /* UNIVERSAL BIT VECTOR */
                        BIT(4096) BASED,
                      CLIST /* CRITERIA LIST */
                        (7) FIXED BIN(15),
                      CMAP /* BIT MAP OF EVENTS IN EACH CLASS*/
                        (0:MAXCL) BIT(B#E) ALIGNED,
                      CV /* UNIVERSAL CHARACTER VECTOR */
                        CHAR(512) BASED,
                      DT /* DENSITY THRESHOLD */
                        FLOAT,
                      E(#E) /* EVENT REPRESENTATIONS */
                        BIT(BLEN) ALIGNED,
                      ECLASS /* THE CLASS OF EACH EVENT */
                        (#E) FIXED BIN(15),
```

```
STMT LEV NT                                        AQ70UNI - VERSION 2

 331    1  0     GET LIST(OLIST);
 332    1  0     GET LIST(NCRIT,(CLIST(I) DO I=1 TO NCRIT));
 333    1  0     GET LIST((TLIST(I) DO I=1 TO NCRIT));

 334    1  0     IF NCRIT < 1 THEN DO;
 335    2  1       NCRIT = 1;
 336    2  1       CLIST(1) = 1;
 337    2  1       TLIST(1) = 0.0;
 338    2  1       END;
 339    1  0     SNCRIT = NCRIT;
 340    1  0     SCLIST = CLIST;
 341    1  0     STLIST = TLIST;

                /*  GET INPUT EVENTS      */
 342    1  0     E = '0'B;
 343    1  0     IF INFORM = 'VECTOR' THEN CALL READVEC;
 344    1  0     ELSE IF INFORM = 'GAMMA' THEN CALL READGAM ;
 345    1  1     ELSE DO ;
 346    2  1       PUT SKIP(3) LIST ('** INVALID EVENT FORMAT SPECIFIED  **');
 347    2  1       FLUSH = '1'B;
 348    2  1       END ;
 349    1  0     IF FLUSH THEN GO TO BLKB_END;
 350    1  0     PUT SKIP(2);

 351    1  0     J1,J2 = 0;
 352    1  1     DO J = 1 TO NCRIT;
 353    2  1       IF CLIST(J)=3 THEN J1=1;
 354    2  1       ELSE IF CLIST(J)=5 THEN J2=1;
 355    2  1       END;
 356    1  0     IF J1=1 THEN DO;
 357    2  1       ALLOCATE Z;
 358    2  1       Z = 1.0;
 359    2  1       GET DATA(Z);
 360    2  1       IF STRACE THEN PUT SKIP DATA(Z);
 361    2  1       END;
 362    1  0     IF J2=1 THEN DO;
 363    2  1       ALLOCATE W;
 364    2  1       W = 1.0;
 365    2  1       GET DATA(W);
 366    2  1       IF STRACE THEN PUT SKIP DATA(W);
 367    2  1       END;

 368    1  0     SPARSEMASK = '0'B;
 369    1  0     DO I = 1 TO NV;
 370    2  1       IF NL(I) < (NC(I)*BPC) THEN
 371    3  2         DO J = BO(I)+NL(I) TO (CO(I)+NC(I)-1)*BPC;
 372    3  2         SUBSTR(SPARSEMASK,J,1) = '1'B;
 373    3  2         END;
 374    2  1       END;

 374    1  0     CHAP = '0'B;
 375    1  0     J = 1;
 377    1  0     DO I = 0 TO MAXCL;
 378    2  1       SUBSTR(CHAP(I),J,NZ(I)) = ONES;
 379    2  1       J = J + NZ(I);
 380    2  1       END;

                /* READ IN UNICLASS COVER OPTIONS */
 381    1  0     UNICLASS=0;
 382    1  0     UNITRACE,OUT,SAVEC='0'B;
 383    1  0     GET DATA (UNICLASS,UNITRACE,OUT,SAVEC,DEBUG);
 384    1  0     IF DEBUG THEN PUT DATA(Z,ZERO,BCLASS,CHAP,SPARSEMASK,
                  ONES,OLIST,CLIST,NCRIT,NCLIST,NCRIT,TLIST,STLIST,
                  SBITS,UNICLASS,UNITRACE,OUT,MAIND,MAVARS,#AVARS);
 385    1  0     IF SAVEC THEN OPEN FILE(COVER) LINESIZE(80) ;
 386    1  0     IF UNITRACE THEN OUT='1'B;
 387    1  0     PUT SKIP EDIT('NUMBER OF CALLS FOR UNICLASS COVER =',
                  UNICLASS)(A,F(4));
 388    1  0     IF UNICLASS<0 THEN DO;
 389    2  1       PUT SKIP LIST('UNICLASS CANNOT BE NEGATIVE. 0 ASSUMED');
 390    2  1       FLUSH = '1'B;
 391    2  1       GO TO BLKB_END;
 392    2  1       END;
 393    1  0     PUT SKIP EDIT('UNITRACE =',UNITRACE,'QUICK UNITRACE =',OUT,
                  'SAVE COVER DATA =',SAVEC) (A,P'(2)',A(5),A,P'(2)',A(5),A,P'(2)');

                /* READ IN UNICLASS COVER OPTIONS */
 398    1  0     ONEUNI:  DO I=1 TO UNICLASS;
 399    2  1       MODE='REL';  DT=0.0; NGB=3;  RANK=NV;  TYPE='IC';
 400    2  1       NCRIT=0;  OLIST=ONES;  ST = NV;
 403    2  1       OP1,OP2 = '0'B;
 404    2  1       GET DATA (MODE,DT,ST,NGB,OLIST,NCRIT,RANK,TYPE,
                    DEBUG,OP1,OP2);
 405    2  1       IF DEBUG THEN UNITRACE,OUT = '1'B;
 406    2  1       IF TYPE ^= 'DC' THEN TYPE = 'IC';
 407    2  1       IF NCRIT > 0 THEN DO;
 408    3  2         DO J = 1 TO NCRIT;
 409    4  2           GET LIST(VAL);
 410    4  2           IF J <= 7 THEN CLIST(J) = VAL;
 411    4  2           ELSE PUT SKIP LIST('CRITERION VALUE',VAL,
                        'NOT USED');
 412    4  2           END;
 413    3  2         DO J = 1 TO NCRIT;
 414    4  2           GET LIST(FVAL);
 415    4  2           IF J <= 7 THEN TLIST(J) = FVAL;
 416    4  2           ELSE PUT SKIP LIST('TOLERANCE VALUE',FVAL,
                        'NOT USED');
 417    4  2           END;
 418    3  2         IF NCRIT > 7 THEN NCRIT = 7;
 419    3  2         END;
 420    2  1       ELSE DO;
```

```
STMT LEV NT                                   AQ7UNI - VERSION 2

                        REVN /* THE EVENT NO WITHIN CLASS FOR EACH EVENT*/
                          (0:E) FIXED BIN(15),
                        ERANKC /* EVENT CHAIN BY RANK */
                          (0:E) FIXED BIN(15),
                        FVAL /* CRITERIA EVALUATION FLOATING RESULT */
                          FLOAT,
                        NCLIST /* DEFAULT CRITERIA LIST */
                          (7) FIXED BIN(15),
                        NNCRIT /* DEFAULT NUMBER OF CRITERIA */
                          FIXED BIN(15),
                        MODE /* EXACT OR APPROX OR REL OR FREE */
                          CHAR(6) VAR ALIGNED,
                      1 SQ /* STORAGE FOR COMPLEXES */
                          (NSC),
                        2 #EE /* NUMBER OF EVENTS COVERED NOT REMAINING */
                          FIXED BIN(31),
                        2 #ER /* NUMBER OF EVENTS COVERED REMAINING */
                          FIXED BIN(31),
                        2 DEN /* DENSITY OF COMPLEX */
                          FLOAT,
                        2 INT /* BIT REPRESENTATION OF COMPLEX */
                          BIT(BLEN) ALIGNED,
                        2 NRANK /* RANK OF COMPLEX */
                          FIXED BIN(31),
                        NTLIST /* DEFAULT TOLERANCE LIST */
                          (7) FLOAT,
                        NCRIT /* NUMBER OF CRITERIA */
                          FIXED BIN(15),
                        NGB /* NUMBER OF NEIGHBORHOODS */
                          FIXED BIN(31),
                        OLIST /* ORDERING INFORMATION */
                          (0:MAXCL) FIXED BIN(15),
                        ONES /* SOURCE OF STRINGS OF ONES */
                          BIT(WLEN) ALIGNED,
                        (OP1,OP2) /* SPECIAL OPTIONS */
                          BIT(1) ALIGNED,
                        RANK /* RANK LIMIT VALUE */
                          FIXED BIN(31),
                        RANKS /* CHAINS OF EVENTS OF SAME RANK */
                          (0:NV+1) FIXED BIN(31),
                        QUT /* QUICK TRACE OPTION */
                          BIT(1) ALIGNED,
                        SAVEC /* SAVE COMPLEXES OPTION */
                          BIT(1) ALIGNED,
                        SBITS /* STRUCTURE BIT REPRESENTATIONS */
                          (#STRUC) BIT(MAXSB) ALIGNED,
                        SBWORK /* STRUCTURE WORK AREA */
                          BIT(MAXSB) ALIGNED,
                        SPAREMASK /* INDICATES UNASSIGNED BITS IN REPS */
                          BIT(BLEN) ALIGNED,
                        ST /* SELECTOR THRESHOLD */
                          FIXED BIN(31),
                        TLIST /* LIST OF TOLERANCE VALUES */
                          (7) FLOAT,
                        TYPE /* TYPE OF COVER (IC,DC) */
                          CHAR(6) VAR ALIGNED,
                        ULIST /* CLASS SELECTION INDICATOR */
                          BIT(MCL) ALIGNED,
                        UNICLASS /* NUMBER OF CHARACTERIZATIONS */
                          FIXED BIN(31),
                        UNITRACE /* DETAILED TRACE OPTION */
                          BIT(1) ALIGNED,
                        VAL /* CRITERIA EVALUATION VALUE */
                          FIXED BIN(31),
                        W /* EVENT WEIGHT VALUES */
                          (0:MAXCL,1:MAXEV) FLOAT CONTROLLED,
                        WEE /* BIT MAP OF COVERED EVENTS NOT IN REMAINING*/
                          BIT(B#E) ALIGNED,
                        WER /* BIT MAP OF COVERED EVENTS REMAINING */
                          BIT(B#E) ALIGNED,
                        WLNE /* BIT MAP OF COMPLEX */
                          BIT(BLEN) ALIGNED,
                        WNEE /* BIT MAP OF EVENTS IN NEIGHBORHOOD */
                          BIT(B#E) ALIGNED,
                        (W1,W2) /* BIT WORK AREA */
                          BIT(WLEN) ALIGNED,
                        Z /* VARIABLE WEIGHTS */
                          (NV) FLOAT CONTROLLED,

                        (I,J,J1,J2) FIXED BIN(31);

311   3   0             ONES = '0'B;
312   3   0             ONES = ¬ ONES;

313   3   0             SBITS = '0'B;
314   3   0             DO D# = 1 TO DOMAINS;
315   3   0               IF DOMDATA(D#).DTYPE = 'S' THEN DO;
316   3   0                 SI = DOMDATA(D#).STRUCIDX;
317   3   0                 J2 = DOMDATA(D#).#LVLS-1;
318   3   0                 DO I = 0 TO J2;
319   3   0                   J1 = 1; J = I;
320   3   0                   DO WHILE (J1 > 0);
321   3   0                     J1,J = STRUC(SI+J);
322   3   0                     IF J>J2 THEN DO;
323   3   0                       SUBSTR(SBITS(SI+I),J-J2,1) = '1'B;
324   3   0                       SUBSTR(SBITS(SI+J),I+1,1) = '1'B;
325   3   0                       END;
326   3   0                     END;
327   3   0                   END;
328   3   0                 END;
329   3   0               END;
330   3   1             END;

                    /*  INITIALIZE VARIOUS VARIABLES AND CONSTANTS   */
```

```
STET LEV NT                                   AQ70NI - VERSION 2

448    #  3          END;
449    #  2        END;
450    #  1      DINITIAL = $EVN / LINITIAL;
451    #  1      IF MODE='REL' THEN DT = DT * DINITIAL;
452    #  1      IF DT > 1.0 THEN DT = 1.0;
453    #  1      HSEED = HER;
454    #  1      REFRESH = '1'B;
455    #  1      SEEDS = '1'B;
456    #  1      IF DEBUG THEN PUT DATA(HER);

457    #  1      DO WHILE (HER);  /* DO WHILE EVENTS REMAIN */
458    #  2        DO I = 1 TO NGB WHILE (SEEDS);
459    #  3          IF (REFRESH(I)) THEN DO;
460    #  4            IF UNITRACE THEN PUT SKIP EDIT('SERVICING NGBR #',I)
                          (A,F(4));
461    #  4            REFRESH(I) = '0'B;
462    #  4            NGBR(I).SEED = NEXTSEED;
463    #  4            IF SEEDS THEN CALL NGBRHD(I); /* BUILD NEW NGBRHOOD */
464    #  4            ELSE REFRESH(I) = '1'B;
465    #  4            END;
466    #  3          END;

467    #  2        NGL = CRITVAL; /* SELECT BEST NGBRHOOD */
468    #  2        IF QUT THEN CALL NGBREPORT;

469    #  2        IF TYPE='IC' THEN DO;
470    #  3          PER = ADDR(ER(NGL));
471    #  3          FB15 = 0;
472    #  3          J = 1;
473    #  3          DO I = 1 TO C#Z;
474    #  4            FC1 = SUBSTR(PER->CV,I,1);
475    #  4            DO WHILE (FB15>0);
476    #  5              K = J+TRIDX(FB15);
477    #  5              ERANKC(K) = RANKS(0);
478    #  5              RANKS(0) = K;
479    #  5              FC1 = TRANSLATE(FC1,TRDROP);
480    #  5              END;
481    #  4            J = J + BPC;
482    #  4            END;
483    #  3          END;

484    #  2        IF SAVEC THEN PUT SKIP FILE(COVER) LIST(NLNE(NGL));
485    #  2        NQ# = NQ# + 1;
486    #  2        NQ(NQ#).INT = NLNE(NGL);
487    #  2        NQ(NQ#).DEN = NGBR(NGL).DEN;
488    #  2        NQ(NQ#).#EE = NGBR(NGL).#EE;
489    #  2        NQ(NQ#).#ER = NGBR(NGL).#ER;
490    #  2        NQ(NQ#).NRANK = NGBR(NGL).NRANK;
491    #  2        REFRESH(NGL) = '1'B;
492    #  2        NER = ¬ ER(NGL);
493    #  2        HER = HER & NER;
494    #  2        DO I = 1 TO NGB;
495    #  3          IF (¬ REFRESH(I)) THEN DO;
496    #  4            NEE = EE(I) & ER(NGL);
497    #  4            EE(I) = EE(I) & NEE;
498    #  4            PEE = ADDR(EE(I));
499    #  4            NGBR(I).#EE = POPULATION(PEE,1,C#Z);
500    #  4            IF NGBR(I).#ER = 0 THEN DO;
501    #  5              REFRESH(I) = '1'B;
502    #  5              IF UNITRACE THEN PUT SKIP EDIT('NGBR',
                            I,'REJECTED - NO REMAINING EVENTS')(A,F(3),X(1),A);
503    #  5              END;
504    #  4            IF (¬ REFRESH(I)) THEN IF TYPE='DC' THEN DO;
505    #  5              WLNE = NLNE(NGL) & NLNE(I);
506    #  5              PWLNE = ADDR(WLNE);
507    #  5              J = 0;
508    #  5              DO K = 1 TO NV WHILE(J=0);
509    #  6                IF NAIRD(K) = ' ' THEN
510    #  6                  IF UNSPEC(SUBSTR(PWLNE->CV,CO(K),NC(K))) THEN;
511    #  6                  ELSE J=1;
512    #  6                END;
513    #  5              IF J=0 THEN DO;
514    #  6                REFRESH(I) = '1'B;
515    #  6                SUBSTR(HSEED,NGBR(I).SEED,1) = '1'B;
516    #  6                SEEDS = '1'B;
517    #  6                IF UNITRACE THEN PUT SKIP EDIT('NGBR',
                              I,'REJECTED - NOT DISJOINT')(A,F(3),X(1),A);
518    #  6                END;
519    #  5              END;
520    #  4            IF (¬ REFRESH(I)) THEN
521    #  5              IF NEE THEN DO;
522    #  5              EE(I) = EE(I) | NEE;
523    #  5              PEE = ADDR(EE(I));
524    #  5              NGBR(I).#EE = POPULATION(PEE,1,C#Z);
525    #  5              END;
526    #  4            END;
527    #  3          END;
          HSEED = HSEED & HER;
          END;
/* PRINT OUT THE UNICLASS COVER */

528    4  1      PUT SKIP(3) EDIT('THE FOLLOWING',NQ#,
529    4  1        ' CARTESIAN COMPLEXES FORM THE UNICLASS COVER')(A,F(4),A);
530    4  1      PUT EDIT (' FOR CLASSES',(CLASS(J) DO J=1 TO #CLASS))
                    (A,(#CLASS)F(4));
          PUT SKIP EDIT('DENSITY OF LEARNING EVENTS IN',
            ' EVENT SPACE IS',DINITIAL,
            ' DENSITY THRESHOLD IS',DT)
531    4  1      (COL(6),A,A,E(10,3,4),A,E(10,3,4));
532    4  1      PUT SKIP;
          CALL RECOVER(0);
/* PREPARE TO WRITE COMMON CHARACTERISTICS */
```

```
STMT LEV NT                                   AQ70NI - VERSION 2

 421    3  2        NCRIT = #NCRIT;
 422    3  2        CLIST = #CLIST;
 423    3  2        TLIST = #TLIST;
 424    3  2        END;

 425    3  1     IF MODE='FREE'  THEN DT=0.000;
 426    3  1     IF MODE='EXACT' THEN DT=1.0;
 427    3  1     PUT SKIP(3) EDIT('{',I,'}',' MODE = ',MODE,'DT =',DT,
                  'ST =',ST,'NGB =',NGB,
                  'RANK =',RANK,'TYPE = ',TYPE,'ULIST = ',ULIST)
                  (A,F(2),A,I(2),A,A(6),X(5),A,X(10,3,0),X(5),A,F(3),
                  X(5),A,F(3),X(5),A,F(3),X(5),A,A(2),X(5),A,B(NCLL));
 428    3  1     PUT SKIP EDIT ('CLIST = ',(CLIST(J) DO J=1 TO NCRIT),
                  'TLIST = ',(TLIST(J) DO J=1 TO NCRIT))
                   (X(6),A,(NCRIT)F(4),X(10),A,(NCRIT)F(8,3));

 429    3  1     IF ULIST THEN;
 430    3  1     ELSE DO;
 431    3  2       PUT SKIP LIST('ULIST SPECIFIES NO CLASSES OF EVENTS');
 432    3  2       GO TO EXIT_COVER;
 433    3  2       END;
 434    3  1     IF NGB < 3 THEN NGB = 3;

 435    3  1     BLKC: BEGIN;

 436    4  1      DCL #CLAZZ /* NUMBER OF CLASSES THIS CHARACTERIZATION*/
                      FIXED BIN(15),
                    #EVN /* NUMBER OF EVENTS THIS CHARACTERIZATION */
                      FIXED BIN(31),
                    CLAZZ /* LIST IF CLASSES */
                      (NCL) FIXED BIN(15),
                    CRIT /* CRITERIA VALUE FOR EACH NEIGHBORHOOD */
                      (NGB) FLOAT,
                    DENSITY /* DENSITY OF LEARNING SET */
                      FLOAT,
                    EE /* EVENTS COVERED BUT REMAINING */
                      (NGB) BIT(B#E) ALIGNED,
                    ER /* EVENTS COVERED REMAINING */
                      (NGB) BIT(B#E) ALIGNED,
                    ELIS /* INDICATES BUT ONE OF THE BETTER NGHBRHOODS*/
                      (NGB) BIT(1) ALIGNED,
                    IDX /* RANDOM INDEX FOR NEXTSEED */
                      FIXED BIN(31) INIT(C#E),
                    MER /* BIT MAP OF REMAINING EVENTS */
                      BIT(B#E) ALIGNED,
                    MERP /* POINTER TO MER */
                      POINTER,
                    NQ# /* NUMBER OF COMPLEXES GENERATED */
                      FIXED BIN(31),
                    MSEED /* BIT MAP OF REMAINING SEED EVENTS */
                      BIT(B#E) ALIGNED,
                    MSEEDP /* POINTER TO MSEED */
                      POINTER,
                    NEE /* EVENTS IN THE NEIGHBORHOOD */
                      (NGB) BIT(B#E) ALIGNED,
                   1 NGBR /* OTHER NEIGHBORHOOD DATA */
                      (NGB),
                   2 #EE /* NUMBER EVENTS IN EE */
                      FIXED BIN(31),
                   2 #ER /* NUMBER EVENTS IN ER */
                      FIXED BIN(31),
                   2 #LNE /* AREA OF COMPLEX */
                      FLOAT,
                   2 #NLE /* NUMBER EVENTS IN NEE */
                      FIXED BIN(31),
                   2 DEN /* DENSITY OF COMPLEX */
                      FLOAT,
                   2 NRANK /* RANK OF THE COMPLEX */
                      FIXED BIN(31),
                   2 SEED /* SEED EVENT FOR THE NEIGHBORHOOD */
                      FIXED BIN(31),
                     NGL /* NUMBER OF THE BEST NEIGHBORHOOD */
                      FIXED BIN(31),
                     NLNE /* THE COMPLEX COVERING THE NEIGHBORHOOD */
                      (NGB) BIT(BLEN) ALIGNED,
                     NUMC /* NUMBER OF CHARS IN VARIABLE REP */
                      FIXED BIN(15),
                     PEE /* POINTER TO EE */
                      POINTER,
                     PER /* POINTER TO ER */
                      POINTER,
                     PLNE /* POINTER TO LNE */
                      POINTER,
                     PNLNE /* POINTER TO NLNE */
                      POINTER,
                     REFRESH /* INDICATES NEIGHBORHOOD SERVICE REQ */
                      (NGB) BIT(1) ALIGNED,
                     SEEDS /* INCICATES THAT SEEDS REMAIN */
                      BIT(1) ALIGNED,
                     STRT /* OFFSET OF VARIABLE IN REP */
                      FIXED BIN(15),

                     (I,J,K,L) FIXED BIN(31);

 437    4  1     MERP = ADDR(MER); MSEEDP = ADDR(MSEED);
 439    4  1     #CLAZZ,NQ#,BANKS(0) = 0;
 440    4  1     MER = '0'B;
 441    4  1     #EVN = 0;
 442    4  1     DO I = 0 TO NAICL;
 443    4  2       IF (SUBSTR(ULIST,I+1,1)) THEN DO;
 444    4  3          #CLAZZ = #CLAZZ+1;
 445    4  3          CLAZZ(#CLAZZ) = ULIST(I);
 446    4  3          MER = MER | CHAR(I);
 447    4  3          #EVN = #EVN + NE(I);
```

```
STMT LEV NT                              AQ7UNI - VERSION 2

                      UBOUND /* UPPER BOUND ON CRITERA VALUES */
                        FLOAT,

                      (C,I,J,K,L,S,N,O) FIXED BIN(31);
  595    5   1      PNLNE = ADDR(WLNE);
  596    5   1      IF OUT THEN PUT SKIP(2) EDIT
                       ('SELECTING BEST NEIGHBORHOOD') (A);
  597    5   1      SOMEFIT=2;   IMIN = 1;
  599    5   1      ELIM = REPRESS;
  600    5   1      DO I=1 TO NCRIT WHILE (SOMEFIT>1);
  601    5   2        IF OUT THEN PUT SKIP EDIT('NOW APPLYING CRIT #',
                         CLIST(I)) (A,F(4));
  602    5   2        FIRST='1'B;
  603    5   3        DO J=1 TO NGB;
  604    5   3          IF ^ELIM(J) THEN DO;
  605    5   4            PNLNE = ADDR(NLNE(J));
  606    5   4            GO TO CCASE(CLIST(I));

  607    5   4            CCASE(1): FC = -NGBR(J).AMT;
  608    5   4            GO TO NN;

  609    5   4            CCASE(2): NLNE=BOOL(PNLNE->BV,SPARESMASK,'1000'B);
  610    5   4            C=0;
  611    5   4            DO K = 1 TO NV;
  612    5   5              O = CO(K); N = NC(K);
  614    5   5              IF (UNSPEC(SUBSTR(PNLNE->CV,O,N))) THEN
  615    5   5                IF (UNSPEC(SUBSTR(PNLNE->CV,O,N))) THEN C=C+1;
  616    5   4              END;
  617    5   4            FC=C;
                       GO TO NN;

  618    5   4            CCASE(3): NLNE=BOOL(PNLNE->BV,SPARESMASK,'1000'B);
  619    5   4            FC=0.0;
  620    5   4            DO K = 1 TO NV;
  621    5   5              O = CO(K); N = NC(K);
  623    5   5              IF (UNSPEC(SUBSTR(PNLNE->CV,O,N))) THEN
  624    5   5                IF (UNSPEC(SUBSTR(PNLNE->CV,O,N))) THEN FC=FC+Z(K);
  625    5   4              END;
                       GO TO NN;

  626    5   4            CCASE(4): FC = 1.0E0 / NGBR(J).AMT;
  627    5   4            GO TO NN;

  628    5   4            CCASE(5): PTR = ADDR(ER(J));
  629    5   4            L=1;   FC=0.0;
  631    5   4            DO K = 1 TO CER;
  632    5   4              FC1 = SUBSTR(PTR->CV,K,1);
  633    5   4              DO WHILE (FB15 > 0);
  634    5   4                N = TRIDX(FB15)+L;
  635    5   4                FC=FC+W(NCLASS(N),REVN(N));
  636    5   4                FC1 = TRANSLATE(FC1,TRDROP);
  637    5   4                END;
  638    5   4              L = L + BPC;
  639    5   4              END;
  640    5   4            GO TO NN;

  641    5   4            CCASE(6): C=0;
  642    5   4            DO K = 1 TO NV;
  643    5   4              L = POPULATION(PNLNE,CO(K),NC(K));
  644    5   4              IF L < NL(K) THEN C=C+L;
  645    5   4              END;
  646    5   4            FC=C;
  647    5   4            GO TO NN;

  648    5   4            CCASE(7): FC=0.0;
  649    5   4            DO K = 1 TO NV;
  650    5   4              O = CO(K); N = NC(K);
  651    5   4              AVER = 0.0;  #R = 0;  L=0;
  652    5   4              DO S = O TO O+N-1;
  653    5   4                FC1 = SUBSTR(PNLNE->CV,S,1);
  654    5   4                DO WHILE (FB15 > 0);
  655    5   4                  #R = #R+1;
  656    5   4                  R(#R) = L+TRIDX(FB15);
  657    5   4                  AVER = AVER+R(#R);
  658    5   4                  FC1 = TRANSLATE(FC1,TRDROP);
  659    5   4                  END;
  660    5   4                L = L+BPC;
  661    5   4                END;
  662    5   4              IF (#R > 0) & (#R < NL(K)) THEN DO;
  663    5   5                AVER = AVER / #R;  #DI = 0.0;
  664    5   5                DO S = 1 TO #R;
  665    5   5                  #DI = #DI + ABS(AVER-R(S));
  666    5   5                  END;
  667    5   5                FC = FC + (#DI / #R);
  668    5   5                END;
  669    5   4              END;
  670    5   4            GO TO NN;

  671    5   4            NN: CRIT(J) = FC;
  672    5   4            IF OUT THEN PUT EDIT('NGB',J,' VALUE IS  ',FC)
  673    5   4              (COL(30),A,F(4),A,E(10,3,4));
  674    5   4            IF FIRST THEN DO;
  675    5   5              IMIN=J;
  676    5   5              CMIN,CMAX=CRIT(J);
  677    5   5              FIRST='0'B;
  678    5   5              END;
  679    5   4            ELSE DO;
  680    5   5              IF CMIN>CRIT(J) THEN DO;
  681    5   6                CMIN=CRIT(J);
  682    5   6                IMIN=J;
  683    5   6                END;
  684    5   5              ELSE IF CMAX<CRIT(J) THEN CMAX=CRIT(J);
  685    5   5              END;
  686    5   4            END;
  687    5   3          END;
```

```
STET LEV ST                              AQ7UNI - VERSION 2

                    IF #Q$>1 THEN DO;
                      WLNE = BQ(1).INT;
                      PWLNE = ADDR(WLNE);
                      DO I = 1 TO NV;
                        START = CO(I);  NUMC = NC(I);  J = 0;
                        DO K = 2 TO #Q$ WHILE(J=0);
                          PLNE = ADDR(BQ(K).INT);
                          IF SUBSTR(PWLNE->CV,START,NUMC) =
                            SUBSTR(PLNE->CV,START,NUMC) THEN J=1;
                        END;
                      IF J=0 THEN DO K=1 TO #Q$;
                        SUBSTR(BQ(K).INT,BO(I),BL(I)) = ONES;
                        END;
                      ELSE SUBSTR(WLNE,BO(I),BL(I)) = ONES;
                      END;
                    PUT SKIP(3) EDIT('THE FOLLOWING SELECTORS ARE COMMON ',
                      'CHARACTERISTICS') (A);
                    BQ(#Q$+1).INT = WLNE;
                    CALL PRCOVER (BQ$+1);

                    WLNE = BOOL(BQ(BQ$+1).INT,SPARESASK,'1000'B);
                    IF WLNE THEN DO;
                      PUT SKIP(3) EDIT('THE FOLLOWING COMPLEXES HAVE COMMON ',
                        'CHARACTERISTICS REMOVED') (A);
                      CALL PRCOVER (-1);
                      END;
                    PUT SKIP(3) EDIT('THE FOLLOWING VARIABLES DISTINGUISH ',
                      'AMONG THE COMPLEXES') (A);
                    DO I = 1 TO BQ$;
                      WLNE = '0'B;
                      DO J = 1 TO BQ$;
                        IF J ^= I THEN WLNE = WLNE | BQ(J).INT;
                        END;
                      WLNE = WLNE & BQ(I).INT;
                      DO J = 1 TO NV;
                        START = BO(J);  NUMC = BL(J);
                        IF SUBSTR(WLNE,START,NUMC) THEN
                          SUBSTR(BQ(I).INT,START,NUMC) = ONES;
                        END;
                      END;
                    CALL PRCOVER(-1);
                    END;
                  IF $NAVARS>0 THEN DO;
                    PUT SKIP(3) EDIT('WARNING:  THE FOLLOWING VARIABLES ARE NOT ',
                      'APPLICABLE TO SOME EVENTS') (A);
                    DO I = 1 TO $NAVARS;
                      PUT EDIT(VARNAME($AVARS(I))) (COL(20),A);
                      END;
                    END;

/* PROCEDURE TO SELECT NEXT SEED EVENT */

                  NEXTSEED: PROC RETURNS (FIXED BIN(31));

                    DCL I FIXED BIN(31);
                    FB15 = 0;
                    DO I = 1 TO C$E WHILE (FB15 = 0);
                      IDX = IDX + 1;
                      IF IDX > C$E THEN IDX = 1;
                      PC1 = SUBSTR(SEEDP->CV,IDX,1);
                      END;
                    SUBSTR(SEEDP->CV,IDX,1) = TRANSLATE(PC1,TRDROP);
                    IF FB15 > 0 THEN I = 1+TRIDX(FB15)+((IDX-1)*BPC);
                    ELSE SEEDS = '0'B;
                    IF UNITRACE THEN IF SEEDS THEN PUT SKIP EDIT
                      ('NEXTSEED IS',I)(A,F(6));
                    ELSE PUT SKIP EDIT('NEXTSEED IS OUT OF SEEDS')(A);
                    RETURN (I);
                    END NEXTSEED;


/*  PROCEDURE TO SELECT BEST NEIGHBORHOOD USING CRITERION LIST */

                    CRITVAL:  PROC RETURNS (FIXED BIN(31));

                      DCL #R /* NUMBER OF REFERENCES */
                          FIXED BIN(31),
                        AVER /* AVERAGE REFERENCE VALUE */
                          FLOAT,
                        CCASE /* CASE LABELS */
                          (7) LABEL,
                        CMAX /* MAXIMUM CRITERIA VALUE */
                          FLOAT,
                        CMIN /* MINIMUM CRITERIA VALUE */
                          FLOAT,
                        FC /* CRITERIA EVALUATION VALUE */
                          FLOAT,
                        FIRST /* INDICATES FIRST NGB EVALUATED */
                          BIT(1) ALIGNED,
                        IMIN /* NGB NUMBER WITH LOWEST VALUE */
                          FIXED BIN(31),
                        MDI /* MAXIMUM DISTANCE VALUE */
                          FLOAT,
                        PER /* POINTER TO ER */
                          POINTER,
                        PWLNE /* POINTER TO WLNE */
                          POINTER,
                        PWLNE /* POINTER TO WLNE */
                          POINTER,
                        R /* REFERENCE VALUES */
                          (NCMAX*BPC) FIXED BIN(15),
                        SOMEFIT /* INDICATES NUMBER OF NGBS TO CONSIDER*/
                          FIXED BIN(31),
```

```
STAT LEV NT                                   AQ7USI - VERSION 2

                        LASTEV = CUREV;
                        WLNE = WLNE | B(CUREV);
                        SUBSTR(WNEE,CUREV,1) = '1'B;
                        W#NEE = W#NEE + 1;
                        ADDED = ADDED + 1;
                        CUREV = ERANKC(CUREV);
                        IF SNGL THEN GO TO ONEONLY;
                        END;
                      ONEONLY:
                        IF (#IVARS+#SVARS) > 0 THEN DO;
                        IF DEBUG THEN PUT SKIP EDIT('BIT MAP BEFORE GENERALIZE ',
                          WLNE) (A,B);
                        CALL GENERALIZE;
                        END;
                      IF DEBUG THEN PUT SKIP EDIT('BIT MAP OF RANK',I,WLNE,WNEE)
                        (A,F(3),X(8),B,X(5),B);
                      IF TIPS=0C THEN DO J = 1 TO #Q# WHILE(NGBOK);
                        W1 = WLNE & #Q(J).INT;
                        NGBOK = '0'B;
                        DO L = 1 TO #V WHILE(¬NGBOK);
                          IF MAIND(L) = ' ' THEN
                            IF ONSPEC(SUBSTR(PV1->CV,CO(L),BC(L))) THEN;
                          ELSE NGBOK='1'B;
                          END;
                        IF UNITRACE THEN IF (¬NGBOK) THEN PUT SKIP EDIT
                          ('NGBR NOT DISJOINT WITH RESPECT TO COMPLEX ',J)
                          (A,F(4));
                        END;
                      W#LNE = 0.0;
                      IF NGBOK & (ST<NV) THEN DO;
                        W#LNE = 1.0;  K = NV;
                        IF W#NEE > 1 THEN DO J = 1 TO NV;
                          L = POPULATION(PWLNE,CO(J),BC(J));
                          IF L>0 THEN W#LNE = W#LNE * L;
                          IF L=NL(J) THEN K = K-1;
                          END;
                        IF K>ST THEN DTCHK='0'B;
                        ELSE DTCHK = DT > 0.0;
                        END;
                      IF NGBOK & DTCHK THEN DO;
                        IF W#LNE < 1.0 THEN DO;
                          W#LNE = 1.0;
                          IF W#NEE > 1 THEN DO J = 1 TO NV;
                            L = POPULATION(PWLNE,CO(J),BC(J));
                            IF L>0 THEN W#LNE = W#LNE * L;
                            END;
                          END;
                        NGBOK = (W#NEE / W#LNE) >= DT;
                        HOPELESS = (#EVN / W#LNE) < DT;
                        IF DEBUG THEN PUT SKIP DATA(W#LNE,NGBOK,HOPELESS,SOMEOK);
                        IF ¬ (NGBOK | HOPELESS) THEN DO;
                          W#EE = W#NEE;  BEE = WNEE;
                          CALL ECOV(I,CUREV,W#EE,WEE,WLNE);
                          W#EE = 0;  WEE = '0'B;
                          IF RANKS(0)>0 THEN CALL ECOV(0,RANKS(0),W#EE,WEE,WLNE);
                          WDEN = (W#EE+W#EE) / W#LNE;
                          NGBOK = WDEN >= DT;
                          IF NGBOK THEN DO;
                            EE(NGB#) = WEE;
                            EE(NGB#) = WEE;
                            N.#EE = W#EE;
                            N.#EE = W#EE;
                            N.LEN = WDEN;
                            UPDATESAVED = LASTUPDATE + 1;
                            END;
                          END;
                        END;
                      IF NGBOK THEN DO;
                        SOMEOK = '1'B;
                        LASTUPDATE = LASTUPDATE + 1;
                        WLNE(NGB#) = WLNE;
                        NEE(NGB#) = WNEE;
                        N.#LNE = W#LNE;
                        N.#NEE = W#NEE;
                        N.NRANK = I;
                        OLDEV = LASTEV;
                        END;
                      ELSE IF (¬ OP1) THEN DO;
                        IF (¬ SNGL) THEN IF ADDED>1 THEN DO;
                          SNGL,NGBOK = '1'B;
                          IF UNITRACE THEN PUT SKIP LIST('STARTING SINGLE MODE');
                          CUREV = RANKS(I); /* START OVER */
                          END;
                        ELSE;
                        ELSE IF (¬ OP2) THEN DO;
                          IF DEBUG THEN PUT SKIP EDIT('SKIPPING OVER EVENT',
                            LASTEV) (A,F(4));
                          IF CUREV=0 THEN NGBOK = SOMEOK;
                          ELSE NGBOK = '1'B;
                          IF OLDEV <= 0 THEN RANKS(I) = CUREV;
                          ELSE ERANKC(OLDEV) = CUREV;
                          ERANKC(LASTEV) = RANKS(I+1);
                          RANKS(I+1) = LASTEV;
                          END;
                        IF NGBOK THEN DO;
                          IF N.NRANK <= 0 THEN GO TO NGBEGIN;
                          WLNE = WLNE(NGB#);     WNEE = NEE(NGB#);
                          W#LNE = N.#LNE;        W#EE = N.#EE;
                          END;
                        END;
                      END;
                    END;
                  IF LASTUPDATE = 0 THEN DO;
                    N.#EE,N.#ER = 1;
```

```
688   5  5            END;
689   5  4          END;
690   5  3        END;

691   5  2        UBOUND=CHIN+(TLIST(I)*(CHAI-CHIN));

692   5  2        IF OUT THEN PUT SKIP DATA (CHIN,CHAI,UBOUND);
693   5  2        SOMEFIT=0;
694   5  2        DO J=1 TO NGB;
695   5  3          IF ¬ELIM(J) THEN DO;
696   5  4            IF CRIT(J)>UBOUND THEN DO;
697   5  5              ELIM(J) = '1'B;
698   5  5              IF UNITRACE THEN PUT EDIT('NGB',J,' ELIMINATED')
                         (X(4),A,F(4),A);
699   5  5            END;
700   5  4            ELSE SOMEFIT = SOMEFIT + 1;
701   5  4            END;
702   5  3          END;
703   5  2        END;
704   5  1      RETURN (IMIN);
705   5  1      END CRITVAL;

           /* PROCEDURE TO GENERATE A NEIGHBORHOOD */

706   4  1      NGBRHD: PROC (NGB#);

707   5  1       DCL ADDED /* NUMBER OF EVENTS ADDED LAST */
                     FIXED BIN(31),
                   CUREV /* NEXT EVENT TO PROCESS */
                     FIXED BIN(31),
                   DTCHK /* INDICATES DENSITY MUST BE CHECKED */
                     BIT(1) ALIGNED,
                   HOPELESS /* INDICATES DENSITY CANNOT BE MET */
                     BIT(1) ALIGNED,
                   LASTEV /* EVENT LAST PROCESSED */
                     FIXED BIN(31),
                   LASTUPDATE /* NUMBER OF LATEST UPDATE */
                     FIXED BIN(31),
                  1 N /* WORKING STORAGE FOR NGBR */
                  2 #EE /* NUMBER EVENTS IN EE */
                     FIXED BIN(31),
                  2 #ER /* NUMBER EVENTS IN ER */
                     FIXED BIN(31),
                  2 #LNE /* AREA OF COMPLEX */
                     FLOAT,
                  2 #NEE /* NUMBER EVENTS IN NEE */
                     FIXED BIN(31),
                  2 DEN /* DENSITY OF COMPLEX */
                     FLOAT,
                  2 NRANK /* RANK OF THE COMPLEX */
                     FIXED BIN(31),
                  2 SEED /* SEED EVENT FOR THE NEIGHBORHOOD */
                     FIXED BIN(31),
                   NGB# /* NUMBER OF NEIGHBORHOOD */
                     FIXED BIN(31),
                   NGBOK /* INDICATES THRESHOLDS SATISFIED */
                     BIT(1) ALIGNED,
                   OLDEV /* EVENT LAST SUCCESSFULLY INCLUDED */
                     FIXED BIN(31),
                   PLNE /* POINTER TO LNE */
                     POINTER,
                   PWLNE /* POINTER TO WLNE */
                     POINTER,
                   (PW1,PW2) /* POINTERS TO W1, W2 */
                     POINTER,
                   SNGL /* INDICATES EVENTS ADDED ONE AT A TIME */
                     BIT(1) ALIGNED,
                   SOMBOK /* INDICATES SOME EVENTS WERE ADDED */
                     BIT(1) ALIGNED,
                   UPDATESAVED /* UPDATE NUMBER WITH SAVED DATA */
                     FIXED BIN(31),
                   W#EE /* WORK VALUE FOR #EE */
                     FIXED BIN(31),
                   W#ER /* WORK VALUE FOR #ER */
                     FIXED BIN(31),
                   W#LNE /* WORK VALUE FOR #LNE */
                     FLOAT,
                   W#NEE /* WORK VALUE FOR #NEE */
                     FIXED BIN(31),
                   WDEN /* WORK VALUE FOR DEN */
                     FLOAT,

                   (I,J,K,L) FIXED BIN(31);

708   5  1       PW1 = ADDR(W1);  PW2 = ADDR(W2);
710   5  1       PWLNE = ADDR(WLNE);
711   5  1       FB15 = 0;   SNGL = '0'B;  DTCHK = DT > 0.0;
714   5  1       N.SEED = NGBR(NGB#).SEED;
715   5  1       CALL BANKCHAIN(N.SEED);  /* BUILD BANK CHAINS */
716   5  1       NGBEGIN:
                   UPDATESAVED = -1;
717   5  1       W#NEE = 1;
718   5  1       WNEE = '0'B;
719   5  1       SUBSTR(WNEE,N.SEED,1) = '1'B;
720   5  1       WLNE = E(N.SEED);
721   5  1       NGBOK = '1'B;
722   5  1       LASTUPDATE,N.NRANK = 0;
723   5  1       DO I = 1 TO BANK WHILE(NGBOK);
724   5  2         CUREV = BANKS(I);  OLDEV = 0;
726   5  2         SOMBOK = '0'B;
727   5  2         DO WHILE (NGBOK & (CUREV>0));
728   5  3           ADDED = 0;
729   5  3           DO WHILE(CUREV>0);
```

```
STMT LEV NT                                   AQ7UNI - VERSION 2

 908    6  1          W1 = ¬ LNE;
 909    6  1          IF SRANK>0 THEN FRANK = RANK+1;
 910    6  1          ELSE FRANK = 0;
 911    6  1          LINK = SEVNT;
 912    6  1          DO I = SRANK TO FRANK;
 913    6  1            DO WHILE (LINK > 0);
 914    6  1              IF ONITRACE THEN PUT EDIT('EVENT#',LINK)(X(2),A,F(4));
 915    6  1              W2 = W1 & R(LINK);
 916    6  1              IF W2 THEN IF ONITRACE THEN PUT EDIT(' NOT COVERED')(A);
 917    6  1              ELSE; ELSE DO;
 918    6  1                COUNT = COUNT+1;
 919    6  1                SUBSTR(HAP,LINK,1) = '1'B;
 920    6  1                IF ONITRACE THEN PUT EDIT(' IS COVERED')(A);
 921    6  1                END;
 922    6  1              LINK = BRANKC(LINK);
 923    6  1              END;
 924    6  2            IF I<FRANK THEN LINK = RANKS(I+1);
 925    6  2            END;
 926    6  1          END ECOV;

                /* PROCEDURE TO GENERALIZE SELECTORS */

 928    5  1        GENERALIZE: PROC;

 929    6  1          DCL FB /* FIRST BIT INDEX*/
                          FIXED BIN(31),
                        FNSH /* LAST CHAR NUMBER */
                          FIXED BIN(31),
                        NB /* NUMBER OF BITS */
                          FIXED BIN(31),
                        PWLNE /* POINTER TO WLNE */
                          POINTER,
                        SI /* STRUCTURE INDEX */
                          FIXED BIN(31),
                        STRT /* STARTING CHAR NUMBER */
                          FIXED BIN(31),

                        (I,J,L,V) FIXED BIN(31);

 930    6  1          PWLNE = ADDR(WLNE);
 931    6  1          FB15 = 0;
 932    6  1          DO I = 1 TO #IVARS;
 933    6  1            V = INTVARS(I);
 934    6  1            L = BO(V);
 935    6  1            STRT = CO(V); FNSH = STRT+NC(V)-1;
 936    6  1            DO J = STRT TO FNSH;
 937    6  1              FC1 = SUBSTR(PWLNE->CV,J,1);
 938    6  1              IF FC1 ¬= LON(1) THEN GO TO FSTBIT;
 939    6  1              L = L + BPC;
 940    6  1              END;
 941    6  1            GO TO NOI;  /* NO BITS ON, SKIP TO NEXT */
 942    6  1            FSTBIT: FB = L+TRIDX(FB15);
 943    6  1            L = ((NC(V)-1) * BPC) + BO(V);
 944    6  1            IF J < FNSH THEN DO J = FNSH TO STRT BY -1;
 945    6  1              FC1 = SUBSTR(PWLNE->CV,J,1);
 946    6  1              IF FC1 ¬= LON(1) THEN GO TO LSTBIT;
 947    6  1              L = L - BPC;
 948    6  1              END;
 949    6  1            LSTBIT: DO WHILE (FB15>0);
 950    6  1              NB = FB15;
 951    6  1              FC1 = TRANSLATE(FC1,TRDROP);
 952    6  1              END;
 953    6  1            NB = (L+TRIDX(NB))-FB+1;
 954    6  1            IF NB > 2 THEN SUBSTR(WLNE,FB,NB) = ONES;
 955    6  1            NOI: END;
 956    6  2          DO I = 1 TO #SVARS;
 957    6  1            V = STRUCVARS(I);
 958    6  1            L,NB = 0;
 959    6  1            SBWORK = ONES;
 960    6  1            SI = DOSDATA(DOS#(V))-STRUCIDX;
 961    6  1            STRT = CO(V); FNSH = STRT+NC(V)-1;
 962    6  1            DO J = STRT TO FNSH;
 963    6  1              FC1 = SUBSTR(PWLNE->CV,J,1);
 964    6  1              DO WHILE (FB15>0);
 965    6  1                SBWORK = SBWORK & SBITS(SI+L+TRIDX(FB15));
 966    6  1                NB = NB+1;
 967    6  1                FC1 = TRANSLATE(FC1,TRDROP);
 968    6  1                END;
 969    6  1              L = L + BPC;
 970    6  1              END;
 971    6  1            IF NB>1 THEN DO;
 972    6  2              J = INDEX(SBWORK,'1'B);
 973    6  2              IF J=0 THEN SUBSTR(WLNE,BO(V),NL(V))=ONES;
 974    6  2              ELSE SUBSTR(WLNE,BO(V),NL(V)) = SBITS(SI+J-1+NL(V));
 975    6  2              END;
 976    6  1            END;
 977    6  1          END GENERALIZE;

                /* PROCEDURE TO ELIMINATE LESS DESIRABLE SELECTORS */

 980    5  1        DROPSEL: PROC (PLNE,D);

 981    6  1          DCL COUNT /* NUMBER OF BITS FOR EACH VARIABLE */
                          (NV) FLOAT,
                        DROPVAR /* VARIABLE NUMBER TO DROP */
                          FIXED BIN(31),
                        MAXFRACTION /* MAX FRACTION OF ONE BITS */
                          FLOAT,
                        PLNE /* POINTER TO LNE */
                          POINTER,
                        PW2 /* POINTER TO W2 */
                          POINTER,
                        SECONDTRY /* INDICATES SECOND ATTEMPT */
```

```
STMT LEV NT                              AQ7UNI - VERSION 2

 832    5  2          N.$LNE,N.DEN = 1.0;
 833                   N.$EE,N.NRANK = 0;
 834                   EE(NGB$)=NEE(NGB$)='0'B;
 835                   SUBSTR(NEE(NGB$),N.SEED,1) = '1'B;
 836                   EE(NGB$) = NEE(NGB$);
 837                   NLNE(NGB$) = E(N.SEED);
 838                   END;
 839                ELSE DO;
 840                   IF (N.$LNE<1.0) | (ST<NV) THEN DO;
 841                     N.$LNE = 1.0;  PLNE = ADDR(NLNE(NGB$));  K = NV;
 842                     IF N.$EE > 1 THEN DO J = 1 TO NV;
 843                       L = POPULATION(PLNE,CO(J),EC(J));
 844                       IF L>0 THEN N.$LNE = N.$LNE * L;
 845                       IF L=NL(J) THEN K = K-1;
 846                       END;
 847                     IF K>ST THEN CALL DROPSEL (PLNE,K-ST);
 848                     END;
 849                   IF LASTUPDATE ^= UPDATESAVED THEN DO;
 850                     N.$EE = N.$NEE;  EE(NGB$) = NEE(NGB$);
 851                     N.$EE = 0;  EE(NGB$) = '0'B;
 852                     IF N.$NEE > 1 THEN DO;
 853                       IF OLDEV>0 THEN OLDEV = ERANKC(OLDEV);
 854                       CALL ECOV(N.NRANK,OLDEV,N.$EE,EE(NGB$),NLNE(NGB$));
 855                       IF RANKS(0)>0 THEN CALL ECOV(0,RANKS(0),N.$EE,EE(NGB$),
 856                         NLNE(NGB$));
 857                       END;
 858                     N.DEN = (N.$EE+N.$EE)/N.$LNE;
 859                     END;
 860                   END;
 861                 NGBR(NGB$) = N;
 862                 IF DEBUG THEN PUT SKIP DATA(NGBR(NGB$),NLNE(NGB$),
 863                   NEE(NGB$),EE(NGB$),EE(NGB$));

               /* PROCEDURE TO CHAIN EVENTS BY RANK */

 864    5  1       RANKCHAIN: PROC (SEED);

 865    6  1         DCL EVENT /* EVENT NUMBER */
                        FIXED BIN(31),
                      SEED /* THE SEED EVENT NUMBER */
                        FIXED BIN(31),

                      (I,J,L,R) FIXED BIN(31);

 866    6  1         FB15 = 0;
 867    6  1         IF #NAVARS>0 THEN DO;
 868    6  2           NLNE = E(SEED);
 869    6  2           DO I=1 TO #NAVARS;
 870    6  3             R = NAVARS(I);  J = BO(R);  L = NL(R);
 871    6  3             IF SUBSTR(NLNE,J,L) THEN;
 872    6  3             ELSE SUBSTR(NLNE,J,L) = ONES;
 873    6  3             END;
 874    6  2           NLNE = ^ NLNE;
 875    6  2           END;
 876    6  1         ELSE NLNE = ^ E(SEED);
 877    6  1         L = 1;
 878    6  1         DO I = 1 TO RANK+1;
 879    6  2           RANKS(I) = 0;
 880    6  2           END;
 881    6  1         DO I = 1 TO C#E;
 882    6  2           FC1 = SUBSTR(SERP->CV,I,1);
 883    6  2           DO WHILE (FB15 > 0);
 884    6  3             EVENT = L+TRIDX(FB15);
 885    6  3             IF EVENT ^= SEED THEN DO;
 886    6  4               N2 = NLNE & E(EVENT);
 887    6  4               R = 0;
 888    6  4               DO J = 1 TO CLEN WHILE (R<=RANK);
 889    6  5                 IF SUBSTR(PN2->CV,J,1) ^= LOW(1) THEN R=R+1;
 890    6  5                 END;
 891    6  4               IF UNITRACE THEN PUT SKIP EDIT('EVENT',EVENT,
 892    6  4                 'IS OF RANK',R) (A,F(4),X(1),A,F(4));
 893    6  4               IF R < 1 THEN R = 1;
 894    6  4               ERANKC(EVENT) = RANKS(R);
 895    6  4               RANKS(R) = EVENT;
 896    6  4               END;
 897    6  3             FC1 = TRANSLATE(FC1,TRDROP);
 898    6  3             END;
 899    6  2           L = L + BPC;
 900    6  2           END;
 901    6  1         IF DEBUG THEN PUT SKIP DATA(RANKS,ERANKC);
 902    6  1         END RANKCHAIN;

               /* PROCEDURE TO FIND COVERED EVENTS */

 903    5  1       ECOV: PROC (SRANK, SEVNT, COUNT, MAP, LNE);

 904    6  1         DCL COUNT /* CURRENT POPULATION COUNT */
                        FIXED BIN(31),
                      LINK /* LINK TO NEXT EVENT ON RANK CHAIN */
                        FIXED BIN(31),
                      LNE /* BIT REPRESENTATION OF COMPLEX */
                        BIT(*) ALIGNED,
                      MAP /* BIT MAP OF EVENTS COVERED */
                        BIT(*) ALIGNED,
                      FRANK /* STOPPING RANK VALUE */
                        FIXED BIN(31),
                      SEVNT /* STARTING EVENT NUMBER */
                        FIXED BIN(31),
                      SRANK /* STARTING RANK VALUE */
                        FIXED BIN(31),

                      I FIXED BIN(31);
```

```
STMT LEV NT                                   AQ7UNI - VERSION 2

                        BIT(1) ALIGNED,
                      (D,I,J,K,L) FIXED BIN(31);

  962    6  1        DO I = 1 TO NV;
  963    6  2          L = POPULATION(PLNE,CO(I),NC(I));
  964    6  2          IF (L<2) | (L=NL(I)) THEN COUNT(I) = 0;
  965    6  2          ELSE COUNT(I) = FLOAT(L) / FLOAT(NL(I));
  966    6  2          END;
  967    6  1        IF DEBUG THEN PUT SKIP DATA(D,COUNT);

  968    6  1        DO SECONDTRY = '0'B, '1'B;
  969    6  2          DROPVAR = 1;
  990    6  2          DO WHILE ((D>0) & (DROPVAR>0));
  991    6  3            MAXFRACTION = 0.0;  DROPVAR = 0;
  993    6  4            DO I = 1 TO NV;
  994    6  4              IF SECONDTRY | (VARTYPE(I)='F') THEN
  995    6  5                IF MAXFRACTION < COUNT(I) THEN DO;
  996    6  5                MAXFRACTION = COUNT(I);
  997    6  5                DROPVAR = I;
  998    6  5                END;
  999    6  4              END;
 1000    6  3            IF DROPVAR > 0 THEN DO;
 1001    6  4              IF TYPE='DC' THEN W1 = PLNE->BV;
 1002    6  4              SUBSTR(PLNE->BV,BO(DROPVAR),NL(DROPVAR)) = ONES;
 1003    6  4              IF TYPE='DC' THEN DO;
 1004    6  5                L = 1;  PW2 = ADDR(W2);
 1005    6  5                DO J = 1 TO HQ# WHILE(L=1);
 1006    6  6                  L = 0;
 1007    6  6                  W2 = PLNE->BV & HQ(J).INT;
 1008    6  6                  DO K = 1 TO NV WHILE(L=0);
 1009    6  7                    IF MAIND(K) = '1'B THEN
 1010    6  7                      IF ONSPEC(SUBSTR(PW2->CV,CO(K),NC(K))) THEN;
 1011    6  7                    ELSE L=1;
 1012    6  6                    END;
 1013    6  5                  END;
 1014    6  4                IF L=0 THEN DO;
 1015    6  6                  SUBSTR(PLNE->BV,1,BLEN) = W1;
 1016    6  6                  COUNT(DROPVAR) = 0.0;
 1017    6  5                  END;
 1018    6  4                END;
 1019    6  4              IF COUNT(DROPVAR) ^= 0 THEN DO;
 1020    6  5                D = D-1;
 1021    6  5                IF ^BRIEF THEN PUT EDIT('VARIABLE',DROPVAR,
 1022    6  6                  'DROPPED') (A,F(4));
 1023    6  4                END;
 1024    6  3              END;
 1025    6  2            END;
 1026    6  1          IF DEBUG THEN PUT SKIP EDIT('BIT MAP AFTER DROPSEL',PLNE->BV)
 1027    5  1            (A,X(5),B(BLEN));
                       END DROPSEL;
                     END NGBRHD;
                /* PROCEDURE TO LIST EVENTS COVERED BY NEIGHBORHOOD */
 1028    4  1        NGBREPORT: PROC;
 1029    5  1         DCL PEE /* POINTER TO EE */
                          POINTER,
                        PER /* POINTER TO ER */
                          POINTER,

                        I FIXED BIN(31);

 1030    5  1         FB15 = 0;
 1031    5  1         PUT SKIP;
 1032    5  1         DO I = 1 TO NGB;
 1033    5  2           PUT SKIP EDIT('NGBR',I,' ') (A,F(3));
 1034    5  2           IF REFRESH(I) THEN PUT EDIT('IS IDLE') (A);
 1035    5  2           ELSE DO;
 1036    5  3             IF I=NGL THEN PUT EDIT(' (COMPLEX' ,(NQ#+1),')') (A,F(3));
 1037    5  3             PUT EDIT('RANK =',NGBR(I).NRANK,'AREA =',NGBR(I).$LNE,
                            'DENSITY =',NGBR(I).DEN)  (COL(25),A,F(3),X(5),A,
                            E(10,3,4),X(5),A,E(10,3,4));
 1038    5  3             PUT SKIP EDIT(NGBR(I).#ER,' EVENTS COVERED IN ER:')
                            (COL(10),F(3),A);
 1039    5  3             PER = ADDR(ER(I));
 1040    5  3             CALL PUTEVTS(PER);
 1041    5  3             PUT SKIP EDIT(NGBR(I).#EE,' EVENTS COVERED IN EE:')
                            (COL(10),F(3),A);
 1042    5  3             PEE = ADDR(EE(I));
 1043    5  2             IF NGBR(I).#EE > 0 THEN CALL PUTEVTS(PEE);
 1044    5  2             END;
 1045    5  1           END;
                /* PROCEDURE TO WRITE AN EVENT LIST */
 1046    5  1        PUTEVTS: PROC (PE);
 1047    6  1         DCL C /* TYPE OF ENTRY (E-EVENT,S-SEED) */
                          CHAR(1) ALIGNED,
                        EV /* EVENT NUMBER */
                          FIXED BIN(31),
                        NO /* COLUMN NUMBER FOR FORMAT */
                          FIXED BIN(31),
                        PE /* POINTER TO EVENT BIT MAP */
                          POINTER,
                        PREVCLASS /* CLASS OF PREVIOUS EVENT */
                          FIXED BIN(31),
                        SEED /* SEED EVENT NUMBER */
                          FIXED BIN(31),
```

```
STMT LEV NT                                   AQ70NI - VERSION 2

                        (J,L) FIXED BIN(31);

1048    6   1           NO = 0; L = 1;
1050    6   1           SEED = NGBR(I).SEED;
1051    6   1           PREVCLASS = -1;
1052    6   1           DO J = 1 TO C#R;
1053    6   2             FC1 = SUBSTR(PR->CV,J,1);
1058    6   2             DO WHILE (FB15 > 0);
1055    6   3               EV = L + TRIDX(FB15);
1056    6   3               IF ECLASS(EV) ^= PREVCLASS THEN DO;
1057    6   4                 PREVCLASS = ECLASS(EV);
1058    6   4                 NO = NO + 1;
1059    6   4                 IF NO > 20 THEN NO = 1;
1060    6   4                 PUT EDIT('C',NUM(PREVCLASS,0)) (COL(32+(NO*4)),A,A);
1061    6   4                 END;
1062    6   3               NO = NO + 1;
1063    6   3               IF NO > 20 THEN NO = 1;
1064    6   3               IF EV = SEED THEN C = 'S';
1065    6   3               ELSE C = 'T';
1066    6   3               PUT EDIT(C,NUM(SEED(EV),0)) (COL(32+(NO*4)),A,A);
1067    6   3               FC1 = TRANSLATE(FC1,TRDROP);
1068    6   3               END;
1069    6   2             L = L + BPC;
1070    6   2             END;
1071    6   1           END FOTEVTS;
1072    5   1         END NGBREPORT;

/* PROCEDURE TO PRINT COMPLEXES */
1073    4   1       PRCOVER: PROC (PARM);
1074    5   1       DCL BV /* BEGINNING INTERVAL VALUE */
                        FIXED BIN(31),
                    EV /* ENDING INTERVAL VALUE */
                        FIXED BIN(31),
                    FNSH /* ENDING CHAR POSITION IN LNE */
                        FIXED BIN(15),
                    LINE /* OUTPUT LINE IMAGE */
                        CHAR(200) VAR ALIGNED,
                    LL /* LENGTH OF OUTPUT LINE */
                        FIXED BIN(31),
                    NI /* NAMES INDEX */
                        FIXED BIN(31),
                    NOPUT /* TRUE WHILE NO OUTPUT PRODUCED */
                        BIT(1) ALIGNED,
                    PARM /* INDICATES COMPLEXES TO PRINT */
                        FIXED BIN(31),
                    PNO /* LAST NO TO PRINT */
                        FIXED BIN(15),
                    PNLNE /* POINTER TO NLNE */
                        POINTER,
                    SI /* STRUCTURES INDEX */
                        FIXED BIN(31),
                    SNO /* STARTING NO NUMBER */
                        FIXED BIN(15),
                    STRT /* FIRST CHAR POSITION IN LNE */
                        FIXED BIN(15),

                        (I,J,K,L,P) FIXED BIN(31);

1075    5   1           FB15,SI,NI = 0;
1076    5   1           IF PARM<0 THEN SNO,PNO = PARM;
1077    5   1           ELSE DO; SNO=1; PNO=NO#; END;
1081    5   1           DO I = SNO TO PNO;
1082    5   2             PNLNE = ADDR(HQ(I).INT);
1083    5   2             IF PARM<=0 THEN PUT SKIP EDIT('COMPLEX',I)(A,F(4));
1084    5   2             IF PARM=0 THEN DO;
1085    5   3               K = HQ(I).#ER + HQ(I).#ER;
1086    5   3               PUT EDIT(' OF RANK',HQ(I).RANK,' COVERS',
                                K,' EVENTS (',HQ(I).#ER,' REM) WITH DENSITY OF',HQ(I).DEN)
                                (A,F(3),A,F(5),A,F(3),A,E(10,3,4));
1087    5   3               END;
1088    5   2             LINE = ' ';  NOPUT = '1'B;
1090    5   2             DO J = 1 TO NV;
1091    5   3               STRT = CC(J);  FNSH = STRT+WC(J)-1;
1093    5   3               P = POPULATION(PNLNE,STRT,WC(J));
1094    5   3               IF P ^= NL(J) THEN DO;
1095    5   4                 LINE = LINE || ' (' || VARNAME(J) || '=';
1096    5   4                 IF P = 0 THEN LINE = LINE || 'N.A.) ';
1097    5   4                 ELSE DO;
1098    5   5                   L = 0;
1099    5   5                   IF DOMAINS > 0 THEN NI=DOMDATA(DOM#(J)).NAMESIDX;
1100    5   5                   IF VARTYPE(J)='F' THEN DO;
1101    5   6                     DO K = STRT TO FNSH;
1102    5   7                       FC1 = SUBSTR(PNLNE->CV,K,1);
1103    5   7                       DO WHILE (FB15 > 0);
1104    5   8                         LINE=LINE || NUM(L+TRIDX(FB15)+L,NI) || ',';
1105    5   8                         FC1 = TRANSLATE(FC1,TRDROP);
1106    5   8                         END;
1107    5   7                       L = L+BPC;
1108    5   7                       END;
1109    5   6                     SUBSTR(LINE,LENGTH(LINE),1) = ')';
1110    5   6                     LINE = LINE || ' ';
1111    5   6                     END;
1112    5   5                   ELSE IF VARTYPE(J)='I' THEN DO;
1113    5   6                     DO K = STRT TO FNSH;
1114    5   7                       FC1 = SUBSTR(PNLNE->CV,K,1);
1115    5   7                       IF FC1 ^= LOW(1) THEN GO TO GETBEG;
1116    5   7                       L = L+BPC;
1117    5   7                       END;
1118    5   6                     GETBEG: BV = L+TRIDX(FB15);
1119    5   6                     L=(WC(J)-1)*BPC;
1120    5   6                     IF K < FNSH THEN DO K = FNSH TO STRT BY -1;
```

```
STAT LEV NT                                   AQ7UNI - VERSION 2

1202    4   4             IF VAL>=0 THEN PUT EDIT(VAL) (COL(35+(7N*3)),F(3));
1203    4   4             ELSE PUT EDIT(' NA') (COL(35+(7N*3)),A);
1204    4   4             END;
1205    4   4           IF VAL >= NL(J) THEN DO;
1206    4   4             VAL = NL(J)-1;
1207    4   4             PUT SKIP LIST('*** VARIABLE',J,
                            'INCORRECTLY SPECIFIED');
1208    4   4             FLUSH = '1'B;
1209    4   4             END;
1210    4   4           IF VAL >= 0 THEN SUBSTR(PE->BV,BO(J)+VAL,1)='1'B;
1211    4   4           ELSE MAIND(J) = '+';
1212    4   4           END;
1213    4   3         END;
1214    4   2       END;
1215    4   1       DO K = 1 TO NV;
1216    4   1         IF MAIND(K) = '+' THEN DO;
1217    4   2           #NAVARS = #NAVARS + 1;
1218    4   2           NAVARS(#NAVARS) = K;
1219    4   2           END;
1220    4   1         END;
1221    4   0       END READVEC;

                    /* PROCEDURE TO READ GAMMA EVENT DATA */
1222    3   0       READGAM: PROC;
1223    4   0         DCL BVAL /* DECODED VARIABLE VALUE */
                          FIXED BIN(31),
                        IE /* EVENT NUMBER */
                          FIXED BIN(31),
                        GAM /* GAMMA VALUE */
                          FIXED BIN(31),
                        PE /* POINTER TO ELEMENT OF E */
                          POINTER,

                          (I,J,K) FIXED BIN(31);
1224    4   0         IE = 0;

1225    4   0         DO K=0 TO MAICL;
1226    4   1           PUT SKIP(2) EDIT('CLASS F(',OLIST(K),') ') (A,F(2),A);
1227    4   1           DO I=1 TO NE(K);
1228    4   2             GET LIST(GAM);
1229    4   2             IF PNTR THEN PUT SKIP EDIT('EVENT NO.',I,'=',GAM)
                            (COL(18),A,F(3),A,COL(35),F(6));
1230    4   2             IE = IE + 1;
1231    4   2             NEVN(IE) = I;   ECLASS(IE) = OLIST(K);
1232    4   2             PE = ADDR(E(IE));
1233    4   2             DO J=NV TO 1 BY -1;
1234    4   3               BVAL=MOD(GAM,NL(J));
1235    4   3               GAM=GAM/NL(J);
1236    4   3               SUBSTR(PE->BV,BO(J)+BVAL,1) = '1'B;
1237    4   3               END;
1238    4   2             END;
1239    4   1           END;
1240    4   0         END READGAM;

1241    3   0       BLKB_END: IF ALLOCATION(H) THEN FREE H;
1242    3   0       IF ALLOCATION(Z) THEN FREE Z;
1243    3   0       END BLKB;

                    /* PROCEDURE TO WRITE A NUMBER OR ASSIGNED SYMBOL */
1244    2   0       NUM: PROC (V,NI) RETURNS (CHAR(30) VAR ALIGNED);
1245    3   0         DCL C /* NUMBER CONSTRUCTION AREA */
                          CHAR(8) ALIGNED,
                        D /* DECIMAL WORK VALUE */
                          FIXED DEC(5),
                        NI /* NAMES INDEX (IF NOT ZERO) */
                          FIXED BIN(31),
                        V /* VALUE TO CONVERT */
                          FIXED BIN(31),

                          I FIXED BIN(31);

1246    3   0         IF NI > 0 THEN RETURN (IDNAMES(NI+V));
1247    3   0         ELSE DO;
1248    3   1           D = V;   C = D;
1249    3   1           DO I = 7 TO 1 BY -1;
1250    3   2             IF SUBSTR(C,I,1) = ' ' THEN GO TO RET;
1251    3   2             END;
1252    3   1           I = 0;
1253    3   1           RET: RETURN (SUBSTR(C,I+1));
1254    3   1           END;
1255    3   0       END NUM;

1256    2   0       BLKA_END: END BLKA;
1257    1   0       IF FLUSH THEN PUT SKIP(2) LIST('SKIPPING TO NEXT PROBLEM');
1258    1   0       INFORM = ' ';
1259    1   1       DO WHILE (INFORM ^= '*');
1260    1   1         GET SKIP EDIT (INFORM) (A(1));
1261    1   1         END;
1262    1   0       GET SKIP;
1263    1   0       PUT PAGE;
1264    1   0       GO TO NEWDATA;
1265    1   0       END AQ7UNI;
```

```
STAT LEV NT                                  AQ70NI - VERSION 2

                                  FC1 = SUBSTR(PNLNE->CV,K,1);
                                  IF FC1 ¬= LOW(1) THEN GO TO GETEND;
                                  L = L-BPC;
                                  END;
                                GETEND: DO WHILE (FB15 > 0);
                                  EV = FB15;
                                  FC1 = TRANSLATE(FC1,TRDROP);
                                  END;
                                EV = L+TRIDX(EV);
                                LINE = LINE || NUM(BV,NI);
                                IF EV > BV THEN LINE = LINE || '..' || NUM(EV,NI);
                                LINE = LINE || ') ';
                                END;
                              ELSE IF VARTYPE(J)='S' THEN DO;
                                SBWORK = ONES;
                                SI = DONDATA(DON#(J)).STRUCIDX;
                                BV = 0;
                                DO K = STRT TO FNSH;
                                  FC1 = SUBSTR(PNLNE->CV,K,1);
                                  DO WHILE (FB15 > 0);
                                    EV = TRIDX(FB15) + L;
                                    BV = BV + 1;
                                    SBWORK = SBWORK & SBITS(SI+EV);
                                    FC1 = TRANSLATE(FC1,TRDROP);
                                    END;
                                  L = L+BPC;
                                  END;
                                IF BV > 1 THEN DO;
                                  EV = INDEX(SBWORK,'1'B);
                                  IF EV=0 THEN LINE = LINE || 'ERROR...';
                                  EV = EV-1+NL(J);
                                  END;
                                LINE = LINE || NUM(EV,NI) || ') ';
                                END;
                              ELSE PUT SKIP LIST('PRCOVER RECEIVES ILLEGAL CODE');
                              END;
                            IF LENGTH(LINE) > 120 THEN DO;
                              PUT SKIP EDIT(LINE)(A(LL));
                              NOPUT = '0'B;
                              LINE = SUBSTR(LINE,LL);
                              END;
                            LL = LENGTH(LINE);
                            END;
                          END;
                        IF LENGTH(LINE) > 1 THEN PUT SKIP EDIT(LINE)(A);
                        ELSE IF NOPUT THEN IF PARM=0 THEN PUT SKIP EDIT
                          (' ( UNIT COMPLEX )') (A);
                        ELSE PUT SKIP EDIT(' ( NONE )') (A);
                        PUT SKIP;
                        END;
                      END PRCOVER;

                    BLKC_END: END BLKC;
                   NEXT_COVER: END CREONI;

                  /* PROCEDURE TO DETERMINE BIT POPULATION COUNT */

                     POPULATION: PROC (P,START,L) RETURNS (FIXED BIN(31));
                      DCL L /* NUMBER OF CHARS IN STRING TO BIT-COUNT */
                            FIXED BIN(15),
                          P /* POINTS TO STRING WHOSE BITS ARE COUNTED */
                            POINTER,
                          POP /* POPULATION COUNT */
                            FIXED BIN(31),
                          START /* CHARACTER OFFSET AT WHICH COUNT STARTS*/
                            FIXED BIN(15),

                          I FIXED BIN(31);

                      POP = 0;  FB15 = 0;
                      DO I = START TO START+L-1;
                        FC1 = SUBSTR(P->CV,I,1);
                        POP = POP + TRPOP(FB15);
                        END;
                      RETURN (POP);
                      END POPULATION;

                  /* PROCEDURE TO READ VECTOR EVENT DATA */

                     READVEC: PROC;

                      DCL IE /* EVENT NUMBER */
                            FIXED BIN(31),
                          PE /* POINTER TO ELEMENT OF E */
                            POINTER,
                          VN /* VARIABLE NUMBER ON OUTPUT LINE */
                            FIXED BIN(31),

                          (I,J,K) FIXED BIN(31);

                      IE = 0;  NAVARS = 0;  #NAVARS = 0;  NAIND=' ';
                      DO K=0 TO MAXCL;
                        PUT SKIP(2) EDIT('CLASS F(',OLIST(K),')') (A,F(2),A);
                        DO I = 1 TO NE(K);
                          IF PNTE THEN PUT SKIP EDIT('EVENT NO.',I,'=')
                            (COL(18),A,F(3),A);
                          IE = IE + 1;
                          PE = ADDR(E(IE));
                          EEVN(IE) = I;  ECLASS(IE) = OLIST(K);  VN = 0;
                          DO J = 1 TO NV;
                            GET LIST(VAL);
                            IF PNTE THEN DO;
                              VN = VN+1;
                              IF VN>27 THEN VN=1;
```

On pages 61 and 62 the actual input stream for four problems is shown. A complete description of the four problems along with interpretations of the results will be found in [Stepp 79].

The first problem is called TRAINS. The data represents ten trains each train consisting of 3 to 5 cars. There are 26 variables defined as follows.

number of cars
no. wheels on car 1
no. wheels on car 2
no. wheels on car 3
no. wheels on car 4
no. wheels on car 5
length of car 1
length of car 2
length of car 3
length of car 4
length of car 5
shape of car 1
shape of car 2
shape of car 3
shape of car 4
shape of car 5
cargo shape--car 1
cargo shape--car 2
cargo shape--car 3
cargo shape--car 4
cargo shape--car 5
cargo amount--car 1
cargo amount--car 2
cargo amount--car 3
cargo amount--car 4
cargo amount--car 5

Six domains are defined:

number of cars: 3 levels	0=3 cars 1=4 cars 2=5 cars
no. wheels: 2 levels	0=2 wheels 1=3 wheels
length of car: 2 levels	0=short 1=long
shape of car: 10 levels	0=open rectangle 1=open trapazoid 2=U-shaped 3=hexagon 4=ellipse 5=double open rectangle 6=closed rectangle 7=jagged top 8=sloping top 9=locomotive
a mixture of 0 1 2	5=open top
a mixture of 3 4 6 7 8	9=closed top

APPENDIX II

Sample Input Stream

Neighborhood judging criteria numbers 6 and 1 will be used to select the best neighborhood during all 18 characterizations performed. Using the ULIST parameter, the first 14 characterizations are made on one class of animals at a time. MODE='FREE' eliminates the density threshold constraint. The four characterizations which follow are characterizations of all animals, without regard to the input class categories. The first two of the four use RANK and selector threshold to determine the degree of generalization. The specifications RANK=3 ST=8 indicates that complexes are to be composed of no more than 8 selectors and that neighborhoods are to be composed of events which differ from the seed event in no more than three variables. The last two characterizations use RANK and density threshold to determine the degree of generalization. In these characterizations, no complex is to have a density less than .05.

In the four characterizations of all animals (the last four) both disjoint and intersecting complexes are obtained for comparison.

The output of the AQ7UNI program corresponding to the input stream reproduced on the following pages is given as Appendix III.

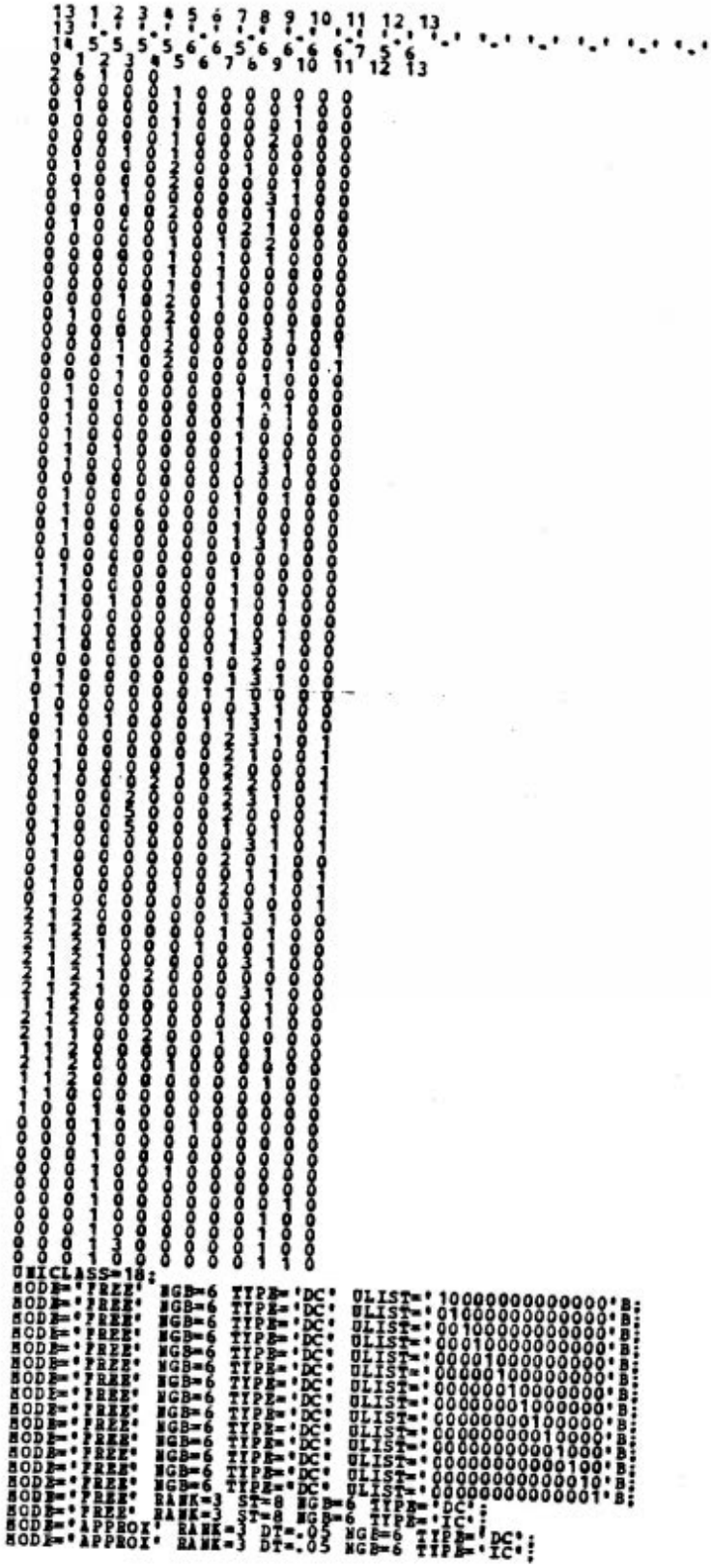
cargo shape:	4 levels	0=circle 1=hexagon 2=triangle 3=rectangle
cargo amount:	4 levels	[0,3]

The ten events are divided into two classes, of 5 events each. Judging criteria number 6 is to be used. When the value of a variable is unknown or not applicable (e.g. the length of car 5 for a train of 3 or 4 cars) the value -1 is given to represent this condition. The characterization parameters for TRAINS are

MODE='APPROX' DT=1E-6 NGB=8 TYPE='DC'

which designate that disjoint complexes are to be produced such that no complex has a density less than 10^{-6}. The selection of best neighborhood is to be made from among 8 neighborhoods built around 8 different, randomly selected seeds. Because the ULIST parameter is not specified, this characterization will be of the union of the two input classes--all ten events.

Skipping the relatively simple second and third problems (BOTTLES and FACES), a few comments are directed to problem four, called ANIMALS. The data describes 79 cute little animals (shown in figure 4), each represented by values of 13 variables. The definitions of the variables can be found at the bottom of page 61. As shown in figure 3, the animals are initially broken into 14 classes. This information is stated to the program in line 3 on page 62.



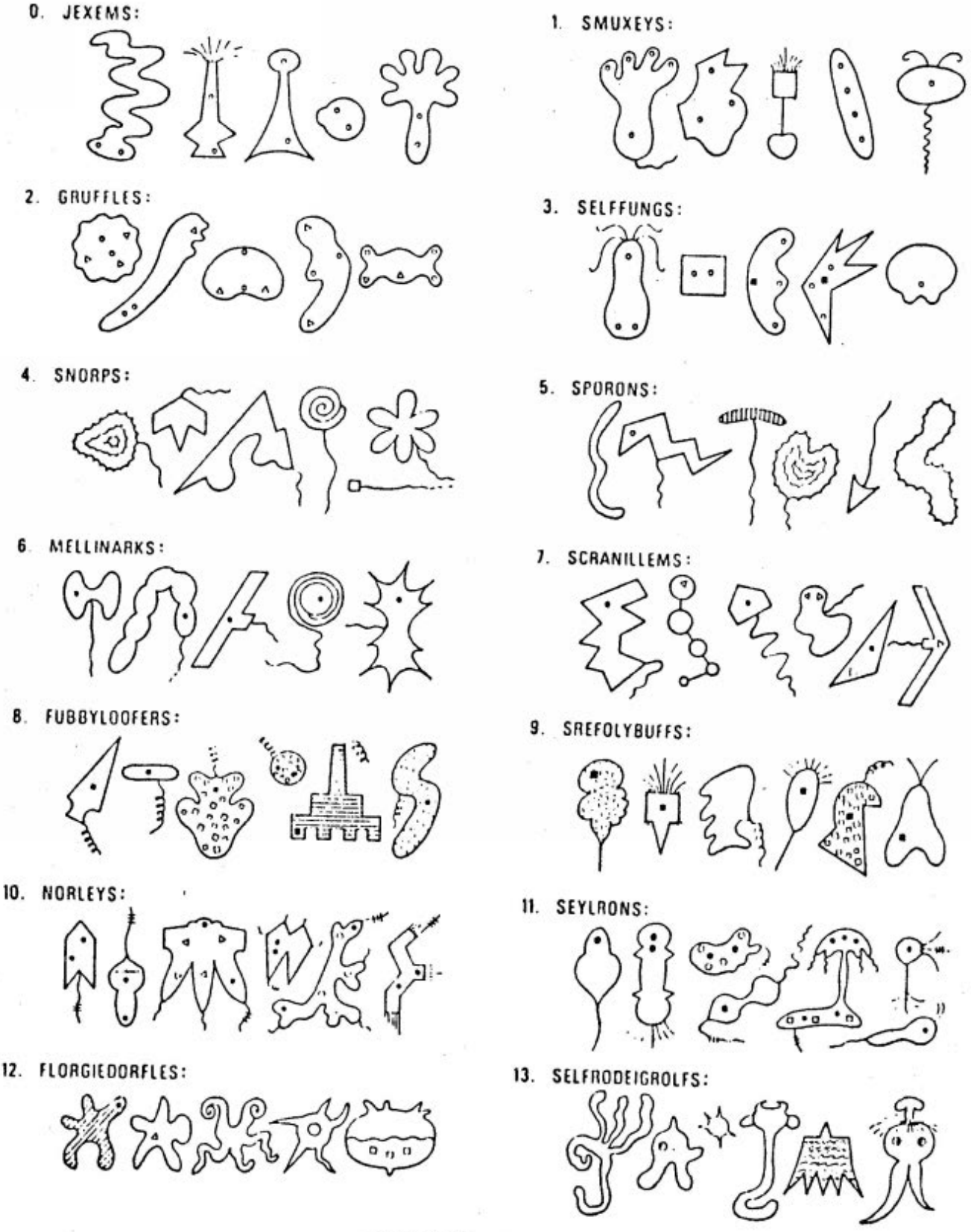
```
INFORM='VECTOR'
NAINV=26 DOMAINS=6  NAMES=24  STLVLS=12
TITLE=2 MAXNAMELEN=16;
 "TRAINS," FROM [LARSON 77], PAGE 107.

 1 'INTERVAL'   3  3 '3' '4' '5'
 2 'INTERVAL:WHEELS'   2  2 '2' '3'
 3 'FACTOR:LENGTH'     2  2 'SHORT' 'LONG'
 4 'STRUCTR:CSHAPE'   10 10 'OPEN RCTNGL' 'OPEN TRAP.' 'U-SHAPED'
                            'HEXAGON' 'ELLIPSE' 'DBL OPEN RCTNGL'
                            'CLOSED RCTNGL' 'JAGGED TOP'
                            'SLOPING TOP' 'LOCOMOTIVE'
                       2  2 'OPEN TOP'  'CLOSED TOP'
                       0  1  2  5
                       6  3  4  6  7  8  9
 5 'FACTOR:LSHAPE'     4  4 'CIRCLE' 'HEXAGON' 'TRIANGLE' 'RECTANGLE'
 6 'INTERVAL:LOAD'  4  0
 26 1 2 2 2 2 2 3 3 3 3 4 4 4 4 4 4 5 5 5 5 5 6 6 6 6 6
 26 'NCARS'  .(1). .(2). .(3). .(4). .(5).
             .(1). .(2). .(3). .(4). .(5).
             .(1). .(2). .(3). .(4). .(5).
             .(1). .(2). .(3). .(4). .(5).
             .(1). .(2). .(3). .(4). .(5).
  2 5 5
  0 1
 1 6 0
2  0  0  0  1  0  1  1  0  1  0  9  0  8  0  0 -1  3  2  1  0  0  3  1  1 -1
1  0  0  0  0 -1 -1  1  0  0  0 -1  9  9  1  6 -1 -1  3  3  3  0 -1  0  1 -1
1  0  0  0  1 -1 -1  1  0  0  1 -1  9  9  0  5 -1 -1  3  3  3  1 -1  0  1 -1
2  0  0  0  0 -1  0  1  0  0  1  0  9  9  1  5 -1 -1  3  3  3  0 -1  0  1 -1
1  0  0  0  1 -1 -1  1  0  0 -1 -1  9  9  0  6 -1 -1  3  3  3  1 -1  0  1 -1
0  0  0  0 -1 -1 -1  1  0 -1 -1 -1  9  9  0  6 -1 -1  3  3  3  1 -1  0  1 -1
1  0  0  0  0 -1 -1  1  0  0 -1 -1  9  9  5  2 -1 -1  3  3  3  1 -1  0  1 -1
0  0  0  1 -1 -1 -1  1  0 -1 -1 -1  9  9  6  2 -1 -1  3  3  3  1 -1  0  1 -1
2  0  0  0  0 -1  0  1  0  0 -1  0  9  9  1  7 -1 -1  3  3  3  0 -1  0  1 -1
0  0  0  0 -1 -1 -1  1  0  1 -1 -1  9  2  0 -1 -1 -1  3  3 -1 -1  0  1  2 -1 -1
UNICLASS=1;
NODE='APPROX' DT=1E-6 NGB=8 TYPE='DC';
*
INFORM='VECTOR'  TITLE=2  NAMES=4  DOMAINS=4 MAXNAMELEN=16;
 "BOTTLES," FROM [NICHALSKI 78], PAGE 26.

1 'INTERVAL'    2 0
2 'INTERVAL'    4 0
3 'INTERVAL'    3 0
4 'INTERVAL'    3 0
4 1 2 3 4
4 '@SQUARES' '@TRIANGLES' '@CIRCLES' '@ASTERISKS'
2 4 4
 0 1
1 6 0
0 1 1 2
1 1 0 2
0 2 1 2
0 1 2 1
1 0 2 1
1 3 0 1
1 2 2 1
1 2 0 1
UNICLASS=3;
NODE='EXACT' ST=2 NGB=8 TYPE='DC';
NODE='EXACT' ST=2 NGB=8 TYPE='IC';
NODE='REF' DT=2 NGB=8 TYPE='DC';
*
 INFORM='VECTOR'  TITLE=2  NAMES=4  DOMAINS=4 MAXNAMELEN=16;
 "FACES," FROM [NICHALSKI 75], FIGURE 3.

 1 'INTERVAL:@CIRCLES'    3 0
 2 'INTERVAL:@OVALS'      3 0
 3 'INTERVAL:@TRIANGLES' 4 0
 4 'INTERVAL:@SQUARES'    3 0
 4 1 2 3 4
 4 '@CIRCLES' '@OVALS' '@TRIANGLES' '@SQUARES'
 2 4 4
  0 1
 1 6 0
2 2 2 0
2 2 2 1
2 2 1 1
2 2 1 2
2 2 0 1
2 3 0 1
2 3 1 1
2 3 1 2
2 3 0 2
UNICLASS=2;
NODE='EXACT' DT=.42 ST=3 NGB=6 TYPE='DC';
NODE='FREE' NGB=6 RANK=2 TYPE='DC';
*
INFORM='VECTOR'  DOMAINS=13  MAXNAMELEN=10
NAMES=40 TITLE=2;
 "ANIMALS," FROM [NICHALSKI 75], EXAMPLE 2.

 1 'INTERVAL:BLK-CIRC'    3 3 '0' '1' '2 OR MORE'
 2 'FACTOR:@TAILS'        2 2 '0' '1 OR MORE'
 3 'INTERVAL:@CROSSHRKS'  3 3 '0' '1 OR 2' '3'
 4 'FACTOR:@EXTERN'       2 2 '0 OR 1' '2 OR MORE'
 5 'FACTOR:TEXTURE'       7 7 'BLANK' 'DOTS' 'HORIZ LINES' 'WAVES'
                              'DIAG LINES' 'COARSE' 'VERT LINES'
 6 'INTERVAL:@ESP-CIRC'   3 3 '0 OR 1' '2' '3 OR MORE'
 7 'FACTOR:@ESP-SQ'       2 2 '0' '1 OR MORE'
 8 'FACTOR:@ESP-TRIANG'   2 2 '0' '1 OR MORE'
 9 'FACTOR:TAIL'          3 3 'NONE' 'STRAIGHT' 'SPRING'
10 'FACTOR:SHAPE'         4 4 'IRREGULAR' 'ELLIPSE' 'CIRCLE' 'TRIANG-SQ'
11 'FACTOR:@ANGLES'       2 2 '0' '1 OR MORE'
12 'FACTOR:@EYES'         2 2 '0' '1 OR MORE'
13 'FACTOR:@BLK-SQ'       2 2 '0' '1 OR MORE'
```



SPECIES OF 'ANIMALS'

Figure 4

from [Michalski 75]

APPENDIX III

Sample Output Listings

```
CLASS F( 0)
                EVENT NO.  1=    2  0  0  0  1  0  1  1  0  0  1  0  9  0  8  0  0 NA  3  2  1  0  0  3  1  1  1
                EVENT NO.  2=    1  0  0  0  0  0  1  1  0  0  0  1  0  2  1  0  6  0 NA  3  2  1  0  0  1  1  2 NA
                EVENT NO.  3=    1  0  0  0  0  1 NA  1  0  0  0  1 NA  9  3  5  6 NA NA  2  2 NA  0  0  1  1  1 NA
                EVENT NO.  4=    2  0  0  0  1  1 NA  1  0  0  0  1 NA  9  3  5  6 NA NA  2  2 NA  0  0  1  1  1 NA
                EVENT NO.  5=    1  0  0  1  0 NA  1  1  0  0  1  0 NA  9  3  5  6 NA NA  2  2  3  0 NA  0  1  1  1
CLASS F( 1)
                EVENT NO.  1=    0  0  0  0 NA NA  1  1  0 NA NA  9  6  0 NA NA NA  0  2 NA NA  0  3  1 NA NA
                EVENT NO.  2=    1  0  0  0  0 NA  1  1  0  0  1 NA  9  5  2  7 NA NA  0  2 NA NA  0  1  1 NA NA
                EVENT NO.  3=    0  0  1  0 NA NA  1  1  0 NA NA  9  5  2 NA NA NA  3  2 NA NA  0  1  1  0 NA
                EVENT NO.  4=    2  0  0  0  0  0  1  1  0  0  1 NA  9  5  2 NA NA NA  3  2 NA NA  0  1  1 NA NA
                EVENT NO.  5=    0  0  0  0 NA NA  1  1  0  1 NA NA  9  1  7  0  1 NA  3  3  3  0  0  1  1  1  1
NUMBER OF CALLS FOR UNICLASS COVER =    1
UNITRACE = 0     QUICK UNITRACE = 0     SAVE COVER DATA = 0

( 1)  MODE = APPROX      BT = 1.000E-06      ST = 26      NGB =  8      RANK = 26      TYPE = DC      ULIST = 11
      CLIST =     6          TLIST =     0.000

THE FOLLOWING   2  CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES   0   1
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 3.104E-14 DENSITY THRESHOLD IS 1.000E-06

COMPLEX   1 OF RANK  7 COVERS     5 EVENTS (  5 NEW) WITH DENSITY OF 2.170E-06  AG=18.8
 (WHEELS(1)=2)  (WHEELS(4)=2)  (WHEELS(5)=2)  (LENGTH(1)=LONG)  (LENGTH(4)=SHORT)  (LENGTH(5)=SHORT)  (CSHAPE(1)=LOCOMOTIVE)
 (CSHAPE(5)=OPEN TRAP.)  (LSHAPE(1)=N.A.)  (LSHAPE(2)=CIRCLE,TRIANGLE,RECTANGLE)  (LSHAPE(3)=CIRCLE,RECTANGLE)
 (LSHAPE(4)=CIRCLE,RECTANGLE)  (LSHAPE(5)=CIRCLE)  (LOAD(1)=0)  (LOAD(2)=1)  (LOAD(3)=1..2)  (LOAD(4)=1..2)  (LOAD(5)=1)

COMPLEX   2 OF RANK 10 COVERS     5 EVENTS (  5 NEW) WITH DENSITY OF 1.929E-06  AG=19.0
 (WHEELS(1)=2)  (WHEELS(2)=2)  (WHEELS(3)=2)  (WHEELS(5)=2)  (LENGTH(1)=LONG)  (LENGTH(3)=SHORT)  (LENGTH(5)=SHORT)
 (CSHAPE(1)=LOCOMOTIVE)  (CSHAPE(5)=OPEN RCTNGL)  (LSHAPE(1)=N.A.)  (LSHAPE(2)=CIRCLE,TRIANGLE,RECTANGLE)
 (LSHAPE(3)=TRIANGLE)  (LSHAPE(4)=HEXAGON,TRIANGLE,RECTANGLE)  (LSHAPE(5)=CIRCLE,RECTANGLE)  (LOAD(1)=0)  (LOAD(2)=1..3)
 (LOAD(3)=1)  (LOAD(4)=0..1)  (LOAD(5)=1)

THE FOLLOWING SELECTORS ARE COMMON CHARACTERISTICS
 (WHEELS(1)=2)  (WHEELS(5)=2)  (LENGTH(1)=LONG)  (LENGTH(5)=SHORT)  (CSHAPE(1)=LOCOMOTIVE)  (LSHAPE(1)=N.A.)
 (LSHAPE(2)=CIRCLE,TRIANGLE,RECTANGLE)  (LOAD(1)=0)  (LOAD(5)=1)

THE FOLLOWING COMPLEXES HAVE COMMON CHARACTERISTICS REMOVED
COMPLEX   1
 (WHEELS(4)=2)  (LENGTH(4)=SHORT)  (CSHAPE(5)=OPEN TRAP.)  (LSHAPE(3)=CIRCLE,RECTANGLE)  (LSHAPE(4)=CIRCLE,RECTANGLE)
 (LSHAPE(5)=CIRCLE)  (LOAD(2)=1)  (LOAD(3)=1..2)  (LOAD(4)=1..2)

COMPLEX   2
 (WHEELS(2)=2)  (WHEELS(3)=2)  (LENGTH(3)=SHORT)  (CSHAPE(5)=OPEN RCTNGL)  (LSHAPE(3)=TRIANGLE)
 (LSHAPE(4)=HEXAGON,TRIANGLE,RECTANGLE)  (LSHAPE(5)=CIRCLE,RECTANGLE)  (LOAD(2)=1..3)  (LOAD(3)=1)  (LOAD(4)=0..1)

THE FOLLOWING VARIABLES DISTINGUISH AMONG THE COMPLEXES
COMPLEX   1
 (CSHAPE(5)=OPEN TRAP.)  (LSHAPE(3)=CIRCLE,RECTANGLE)

COMPLEX   2
 (CSHAPE(5)=OPEN RCTNGL)  (LSHAPE(3)=TRIANGLE)
```

"TRAINS," FROM [LARSON 77], PAGE 107.

```
AQ7UNI - VERSION 2 - OCT 1978
INPUT FORMAT IS VECTOR
DOMAIN DEFINITIONS

   DOMAIN  1 OF INTERVAL TYPE HAS     3 LEVELS     VALUES WERE NAMED AS FOLLOWS:
              0       3
              1       4
              2       5

   DOMAIN  2 OF INTERVAL TYPE HAS     2 LEVELS     VALUES WERE NAMED AS FOLLOWS:
              0       2
              1       3

   DOMAIN  3 OF FACTOR TYPE HAS     2 LEVELS     VALUES WERE NAMED AS FOLLOWS:
              0       SHORT
              1       LONG

   DOMAIN  4 OF STRUCTURE TYPE HAS    10 LEVELS       STRUCTURE MAP FOLLOWS
          OPEN RCTNGL -> OPEN TOP
          OPEN TRAP. -> OPEN TOP
          U-SHAPED -> OPEN TOP
          HEXAGON -> CLOSED TOP
          ELLIPSE -> CLOSED TOP
          DBL OPEN RCTNGL -> OPEN TOP
          CLOSED RCTNGL -> CLOSED TOP
          JAGGED TOP -> CLOSED TOP
          SLOPING TOP -> CLOSED TOP
          LOCOMOTIVE -> CLOSED TOP

   DOMAIN  5 OF FACTOR TYPE HAS     4 LEVELS     VALUES WERE NAMED AS FOLLOWS:
              0       CIRCLE
              1       HEXAGON
              2       TRIANGLE
              3       RECTANGLE

   DOMAIN  6 OF INTERVAL TYPE HAS     4 LEVELS

NUMBER OF VARIABLES =  26
DOMAIN NUMBER FOR EACH VARIABLE:      1  2  2  2  2  2  3  3  3  3  3  4  4  4  4  4  5  5  5  5  5  6  6  6  6  6
NUMBER OF LEVELS FOR EACH VARIABLE:   3  2  2  2  2  2  2  2  2  2  2 10 10 10 10 10  4  4  4  4  4  4  4  4  4  4
NUMBER OF EVENTS SPECIFIED FOR EACH CLASS:
       CLASS    #EVENTS
         0          5
         1          5
```

```
( 2)  MODE = EXACT      DT = 1.000E+00      ST =  2     NGB =  8     RANK =  4     TYPE = IC     ULIST = 11
      CLIST =     6          TLIST =      0.000

THE FOLLOWING   2  CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES   0   1
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 1.111E-01 DENSITY THRESHOLD IS 1.000E+00

COMPLEX   1 OF RANK  2 COVERS     4 EVENTS (  4 NEW) WITH DENSITY OF 3.333E-01   AG=1.56
 (#SQUARES=1) (#ASTERISKS=1)

COMPLEX   2 OF RANK  2 COVERS     6 EVENTS (  4 NEW) WITH DENSITY OF 2.500E-01   AG=2.00
 (#TRIANGLES=1..2) (#ASTERISKS=1..2)


THE FOLLOWING SELECTORS ARE COMMON CHARACTERISTICS
 ( NONE )


( 3)  MODE = REL        DT = 2.000E+00      ST =  4     NGB =  8     RANK =  4     TYPE = DC     ULIST = 11
      CLIST =     6          TLIST =      0.000

THE FOLLOWING   3  CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES   0   1
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 1.111E-01 DENSITY THRESHOLD IS 2.222E-01

COMPLEX   1 OF RANK  2 COVERS     4 EVENTS (  4 NEW) WITH DENSITY OF 3.333E-01   AG=1.56
 (#SQUARES=1) (#ASTERISKS=1)

COMPLEX   2 OF RANK  3 COVERS     3 EVENTS (  3 NEW) WITH DENSITY OF 3.750E-01   AG=1.42
 (#TRIANGLES=1..2) (#CIRCLES=0..1) (#ASTERISKS=2)

COMPLEX   3 OF RANK  0 COVERS     1 EVENTS (  1 NEW) WITH DENSITY OF 1.000E+00   AG=0.00
 (#SQUARES=0) (#TRIANGLES=1) (#CIRCLES=2) (#ASTERISKS=1)


THE FOLLOWING SELECTORS ARE COMMON CHARACTERISTICS
 ( NONE )
```

"BOTTLES," FROM [MICHALSKI 78], PAGE 26.

```
AQ7UNI - VERSION 2 - OCT 1978
INPUT FORMAT IS VECTOR
DOMAIN DEFINITIONS

   DOMAIN  1 OF INTERVAL TYPE HAS    2 LEVELS

   DOMAIN  2 OF INTERVAL TYPE HAS    4 LEVELS

   DOMAIN  3 OF INTERVAL TYPE HAS    3 LEVELS

   DOMAIN  4 OF INTERVAL TYPE HAS    3 LEVELS


NUMBER OF VARIABLES =   4
DOMAIN NUMBER FOR EACH VARIABLE:       1  2  3  4
NUMBER OF LEVELS FOR EACH VARIABLE:    2  4  3  3
NUMBER OF EVENTS SPECIFIED FOR EACH CLASS:
       CLASS    #EVENTS
         0          4
         1          4

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

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

NUMBER OF CALLS FOR UNICLASS COVER =   3
UNITRACE = 0     QUICK UNITRACE = 0     SAVE COVER DATA = 0

( 1)  MODE = EXACT      DT = 1.000E+00      ST =  2     NGB =  8     RANK =  4     TYPE = DC     ULIST = 11
      CLIST =   6            TLIST =    0.000

THE FOLLOWING   3  CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES   0  1
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 1.111E-01 DENSITY THRESHOLD IS 1.000E+00

COMPLEX   1 OF RANK   1 COVERS     4 EVENTS (  4 NEW) WITH DENSITY OF 3.333E-01   AG=1.56
  (#SQUARES=1) (#ASTERISKS=1)

COMPLEX   2 OF RANK   2 COVERS     3 EVENTS (  3 NEW) WITH DENSITY OF 1.667E-01   AG=2.58
  (#SQUARES=0) (#TRIANGLES=1..2)

COMPLEX   3 OF RANK   0 COVERS     1 EVENTS (  1 NEW) WITH DENSITY OF 1.000E+00   AG=0
  (#SQUARES=1) (#TRIANGLES=1) (#CIRCLES=0)  (#ASTERISKS=2)
```

```
COMPLEX    2 OF RANK  2 COVERS     2 EVENTS (  2 NEW) WITH DENSITY OF 3.333E-01
 (#CIRCLES=2) (#TRIANGLES=2..3) (#SQUARES=0)                                      AG=1.59

COMPLEX    3 OF RANK  2 COVERS     2 EVENTS (  2 NEW) WITH DENSITY OF 2.500E-01
 (#CIRCLES=1..2) (#OVALS=2) (#SQUARES=2)                                          AG=2.00

COMPLEX    4 OF RANK  0 COVERS     1 EVENTS (  1 NEW) WITH DENSITY OF 1.000E+00
 (#CIRCLES=1) (#OVALS=2) (#TRIANGLES=1) (#SQUARES=1)                              AG=0.00


THE FOLLOWING SELECTORS ARE COMMON CHARACTERISTICS
 ( NONE )


THE FOLLOWING VARIABLES DISTINGUISH AMONG THE COMPLEXES
COMPLEX    1
 ( NONE )

COMPLEX    2
 ( NONE )

COMPLEX    3
 ( NONE )

COMPLEX    4
 ( NONE )


( 2)  MODE = FREE      DT = 0.000E+00      ST =  4     NGB =  6     RANK =  2     TYPE = DC     ULIST = 11
      CLIST =      6      TLIST =      0.000

THE FOLLOWING   2  CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES   0   1
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 7.407E-02 DENSITY THRESHOLD IS 0.000E+00

COMPLEX    1 OF RANK  2 COVERS     6 EVENTS (  6 NEW) WITH DENSITY OF 1.667E-01
 (#CIRCLES=2)                                                                     AG=2.58

COMPLEX    2 OF RANK  2 COVERS     2 EVENTS (  2 NEW) WITH DENSITY OF 3.333E-01
 (#CIRCLES=1) (#OVALS=2) (#TRIANGLES=1..3) (#SQUARES=1..2)                        AG=1.59


THE FOLLOWING SELECTORS ARE COMMON CHARACTERISTICS
 ( NONE )


THE FOLLOWING VARIABLES DISTINGUISH AMONG THE COMPLEXES
COMPLEX    1
 (#CIRCLES=2)

COMPLEX    2
 (#CIRCLES=1)
```

"FACES," FROM [MICHALSKI 75], FIGURE 3.

```
AQ7UNI - VERSION 2 - OCT 1978
INPUT FORMAT IS VECTOR
DOMAIN DEFINITIONS

   DOMAIN  1 OF INTERVAL TYPE HAS     3 LEVELS

   DOMAIN  2 OF INTERVAL TYPE HAS     3 LEVELS

   DOMAIN  3 OF INTERVAL TYPE HAS     4 LEVELS

   DOMAIN  4 OF INTERVAL TYPE HAS     3 LEVELS


NUMBER OF VARIABLES =   4
DOMAIN NUMBER FOR EACH VARIABLE:        1  2  3  4
NUMBER OF LEVELS FOR EACH VARIABLE:     3  3  4  3
NUMBER OF EVENTS SPECIFIED FOR EACH CLASS:
       CLASS    #EVENTS
         0          4
         1          4

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

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

NUMBER OF CALLS FOR UNICLASS COVER =   2
UNITRACE = 0      QUICK UNITRACE = 0     SAVE COVER DATA = 0

( 1)  MODE = EXACT       DT = 1.000E+00      ST =  3     NGD =  6     RANK =  4     TYPE = DC     ULIST = 11
      CLIST =     6           TLIST =      0.000

THE FOLLOWING   4  CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES   0   1
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 7.407E-02 DENSITY THRESHOLD IS 1.000E+00

COMPLEX   1 OF RANK  1 COVERS    3 EVENTS (  3 NEW) WITH DENSITY OF 7.500E-01   AG=0.41
 (#CIRCLES=2)  (#OVALS=2)  (#SQUARES=1)
```

```
DOMAIN  9 OF FACTOR TYPE HAS     3 LEVELS     VALUES WERE NAMED AS FOLLOWS:
            0       NONE
            1       STRAIGHT
            2       SPRING

DOMAIN 10 OF FACTOR TYPE HAS     4 LEVELS     VALUES WERE NAMED AS FOLLOWS:
            0       IRREGULAR
            1       ELLIPSE
            2       CIRCLE
            3       TRIANG-SQ

DOMAIN 11 OF FACTOR TYPE HAS     2 LEVELS     VALUES WERE NAMED AS FOLLOWS:
            0       0
            1       1 OR MORE

DOMAIN 12 OF FACTOR TYPE HAS     2 LEVELS     VALUES WERE NAMED AS FOLLOWS:
            0       0
            1       1 OR MORE

DOMAIN 13 OF FACTOR TYPE HAS     2 LEVELS     VALUES WERE NAMED AS FOLLOWS:
            0       0
            1       1 OR MORE
NUMBER OF VARIABLES =   13
DOMAIN NUMBER FOR EACH VARIABLE:      1  2  3  4  5  6  7  8  9 10 11 12 13
NUMBER OF LEVELS FOR EACH VARIABLE:   3  3  3  2  7  3  2  2  3  4  2  2  2
NUMBER OF EVENTS SPECIFIED FOR EACH CLASS:
      CLASS     EVENTS
        0          5
        1          5
        2          5
        3          5
        4          6
        5          6
        6          5
        7          6
        8          6
        9          6
       10          6
       11          7
       12          5
       13          6

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

"ANIMALS," FROM [MICHALSKI 75], EXAMPLE 2.

```
AQ7UNI - VERSION 2 - OCT 1978
INPUT FORMAT IS VECTOR
DOMAIN DEFINITIONS

   DOMAIN  1 OF INTERVAL TYPE HAS    3 LEVELS    VALUES WERE NAMED AS FOLLOWS:
             0       0
             1       1
             2       2 OR MORE

   DOMAIN  2 OF FACTOR TYPE HAS      2 LEVELS    VALUES WERE NAMED AS FOLLOWS:
             0       0
             1       1 OR MORE

   DOMAIN  3 OF INTERVAL TYPE HAS    3 LEVELS    VALUES WERE NAMED AS FOLLOWS:
             0       0
             1       1 OR 2
             2       3

   DOMAIN  4 OF FACTOR TYPE HAS      2 LEVELS    VALUES WERE NAMED AS FOLLOWS:
             0       0 OR 1
             1       2 OR MORE

   DOMAIN  5 OF FACTOR TYPE HAS      7 LEVELS    VALUES WERE NAMED AS FOLLOWS:
             0       BLANK
             1       DOTS
             2       HORIZ LINE
             3       WAVES
             4       DIAG LINES
             5       COARSE
             6       VERT LINES

   DOMAIN  6 OF INTERVAL TYPE HAS    3 LEVELS    VALUES WERE NAMED AS FOLLOWS:
             0       0 OR 1
             1       2
             2       3 OR MORE

   DOMAIN  7 OF FACTOR TYPE HAS      2 LEVELS    VALUES WERE NAMED AS FOLLOWS:
             0       0
             1       1 OR MORE

   DOMAIN  8 OF FACTOR TYPE HAS      2 LEVELS    VALUES WERE NAMED AS FOLLOWS:
             0       0
             1       1 OR MORE
```

```
CLASS F(10)
                EVENT NO.  1=    2  1  2  0  0  0  0  0  1  3  1  0  0
                EVENT NO.  2=    2  1  2  0  0  0  0  0  1  0  1  0  0
                EVENT NO.  3=    2  1  2  0  0  0  0  0  0  0  1  0  0
                EVENT NO.  4=    2  1  2  1  0  2  0  0  0  3  1  0  0
                EVENT NO.  5=    2  1  2  1  0  0  0  0  0  3  0  0  0
                EVENT NO.  6=    2  1  2  1  0  0  0  0  0  3  1  0  0
CLASS F(11)
                EVENT NO.  1=    1  1  2  0  0  0  0  0  1  0  1  0  0
                EVENT NO.  2=    2  1  2  1  0  0  0  0  1  0  1  0  0
                EVENT NO.  3=    2  1  1  2  0  2  0  0  0  0  1  0  0
                EVENT NO.  4=    2  1  1  2  0  0  0  0  0  0  1  0  0
                EVENT NO.  5=    2  1  1  2  0  0  0  1  0  0  1  0  0
                EVENT NO.  6=    2  1  1  2  0  0  0  0  0  0  1  0  0
                EVENT NO.  7=    1  1  2  0  0  0  0  0  0  0  1  0  0
CLASS F(12)
                EVENT NO.  1=    1  0  0  1  0  0  0  0  0  0  0  0  0
                EVENT NO.  2=    0  0  0  1  0  0  0  1  0  0  0  0  0
                EVENT NO.  3=    0  0  0  1  0  0  0  0  0  0  0  0  0
                EVENT NO.  4=    0  0  0  1  0  0  1  0  0  0  0  0  0
CLASS F(13)
                EVENT NO.  1=    0  0  0  1  0  0  0  0  0  0  1  0  0
                EVENT NO.  2=    0  0  0  1  0  0  0  0  0  0  1  0  0
                EVENT NO.  3=    0  0  0  1  0  0  0  0  0  0  1  0  0
                EVENT NO.  4=    0  0  0  1  0  0  0  0  0  0  1  0  0
                EVENT NO.  5=    0  0  0  1  3  0  0  0  0  0  1  0  0
                EVENT NO.  6=    0  0  0  1  0  0  0  0  0  0  1  0  0

NUMBER OF CALLS FOR UNICLASS COVER =  18
UNITRACE = 0     QUICK UNITRACE = 0     SAVE COVER DATA = 0

( 1)  MODE = FREE          DT = 0.000E+00          ST = 13          NGB =  6     RANK = 13     TYPE = DC     ULIST = 10000000000000
      CLIST =    6    1                TLIST =    0.000    0.000

THE FOLLOWING   1 CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES   0
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 1.722E-05 DENSITY THRESHOLD IS 0.000E+00
COMPLEX   1 OF RANK  2 COVERS     5 EVENTS (  5 NEW) WITH DENSITY OF 3.125E-01   AG=1.68
 (BLK-CIRC=0) (#CROSSHRKS=0) (TEXTURE=BLANK) (#EMP-CIRC=2) (#EMP-SQ=0) (#EMP-TRIAN=0) (TAIL=NONE)
 (SHAPE=IRREGULAR,CIRCLE) (#EYES=0) (#BLK-SQ=0)

( 2)  MODE = FREE          DT = 0.000E+00          ST = 13          NGB =  6     RANK = 13     TYPE = DC     ULIST = 01000000000000
      CLIST =    6    1                TLIST =    0.000    0.000

THE FOLLOWING   1 CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES   1
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 1.722E-05 DENSITY THRESHOLD IS 0.000E+00
COMPLEX   1 OF RANK  5 COVERS     5 EVENTS (  5 NEW) WITH DENSITY OF 2.315E-02   AG=5.43
 (BLK-CIRC=0) (#CROSSHRKS=0) (TEXTURE=BLANK) (#EMP-SQ=0) (#EMP-TRIAN=0) (SHAPE=IRREGULAR,ELLIPSE,TRIANG-SQ) (#EYES=0)
 (#BLK-SQ=0)
```

CLASS F(2)

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

CLASS F(3)

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

CLASS F(4)

EVENT NO. 1=	0	1	0	0	0	0	0	0	1	0	1	0	0
EVENT NO. 2=	0	1	0	0	1	0	0	0	1	1	1	0	0
EVENT NO. 3=	0	1	0	0	1	0	0	0	1	1	1	0	0
EVENT NO. 4=	0	1	0	0	1	0	0	0	1	1	0	0	0
EVENT NO. 5=	0	1	0	0	1	0	0	0	1	1	0	0	0
EVENT NO. 6=	0	1	0	0	1	0	0	0	1	3	1	0	0

CLASS I(5)

EVENT NO. 1=	0	0	0	0	0	0	0	0	1	0	1	0	0
EVENT NO. 2=	0	1	0	0	0	0	0	0	1	1	1	0	0
EVENT NO. 3=	0	1	0	0	6	0	0	0	1	1	0	0	0
EVENT NO. 4=	0	1	0	0	0	0	0	0	1	1	0	0	0
EVENT NO. 5=	0	1	0	0	0	0	0	0	1	3	1	0	0
EVENT NO. 6=	0	0	0	0	0	0	0	0	0	0	0	0	0

CLASS F(6)

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

CLASS F(7)

EVENT NO. 1=	1	1	0	0	0	0	0	0	1	3	1	0	0
EVENT NO. 2=	0	0	0	0	0	0	0	0	1	2	0	0	0
EVENT NO. 3=	1	1	0	0	0	0	0	0	1	3	1	0	0
EVENT NO. 4=	0	1	0	0	0	0	0	0	1	1	0	0	0
EVENT NO. 5=	1	0	0	0	0	0	0	0	1	0	3	1	0	0
EVENT NO. 6=	0	1	0	1	0	0	0	1	1	3	1	0	0

CLASS F(8)

EVENT NO. 1=	0	1	0	0	0	0	0	0	2	3	1	0	1
EVENT NO. 2=	0	1	0	0	0	0	0	0	2	1	0	0	1
EVENT NO. 3=	0	1	0	0	0	0	1	0	2	0	0	0	1
EVENT NO. 4=	0	1	0	0	0	2	0	0	2	2	0	0	1
EVENT NO. 5=	0	1	0	0	2	0	0	0	2	3	0	0	1
EVENT NO. 6=	0	1	0	0	5	0	0	0	2	0	0	0	1

CLASS F(9)

EVENT NO. 1=	0	1	0	0	5	0	0	0	1	0	1	0	1
EVENT NO. 2=	0	1	0	0	0	0	0	0	1	3	1	0	1
EVENT NO. 3=	0	1	0	0	0	0	0	0	2	0	1	0	0
EVENT NO. 4=	0	1	0	0	0	0	0	0	2	0	1	0	1
EVENT NO. 5=	0	1	0	0	0	0	0	0	2	0	1	0	1
EVENT NO. 6=	0	1	0	0	0	0	0	0	2	0	0	0	1

```
( 8)  MODE = FREE      DT = 0.000E+00      ST = 13      NGB =  6     RANK = 13     TYPE = DC     ULIST = 00000001000000
      CLIST =  6   1                TLIST =  0.000  0.000

THE FOLLOWING   1 CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES   7
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 2.067E-05 DENSITY THRESHOLD IS 0.000E+00
COMPLEX   1 OF RANK  6 COVERS    6 EVENTS (  6 NEW) WITH DENSITY OF 3.125E-02   AG=5.00
 (BLK-CIRC=0,,1) (#CROSSHRKS=0) (TEXTURE=BLANK) (#EMP-CIRC=0 OR 1) (#EMP-SQ=0) (TAIL=NONE,STRAIGHT)
 (SHAPE=IRREGULAR,CIRCLE,TRIANG-SQ) (#EYES=0) (#BLK-SQ=0)


( 9)  MODE = FREE      DT = 0.000E+00      ST = 13      NGB =  6     RANK = 13     TYPE = DC     ULIST = 00000000100000
      CLIST =  6   1                TLIST =  0.000  0.000

THE FOLLOWING   1 CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES   8
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 2.067E-05 DENSITY THRESHOLD IS 0.000E+00
COMPLEX   1 OF RANK  3 COVERS    6 EVENTS (  6 NEW) WITH DENSITY OF 4.167E-02   AG=4.58
 (BLK-CIRC=0) (#TAILS=1 OR MORE) (#CROSSHRKS=0) (#EXTREM=0 OR 1) (TEXTURE=BLANK,HORIZ LINE,COARSE) (#EMP-TRIAN=0)
 (TAIL=SPRING) (#EYES=0) (#BLK-SQ=1 OR MORE)


(10)  MODE = FREE      DT = 0.000E+00      ST = 13      NGB =  6     RANK = 13     TYPE = DC     ULIST = 00000000010000
      CLIST =  6   1                TLIST =  0.000  0.000

THE FOLLOWING   1 CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES   9
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 2.067E-05 DENSITY THRESHOLD IS 0.000E+00
COMPLEX   1 OF RANK  3 COVERS    6 EVENTS (  6 NEW) WITH DENSITY OF 4.167E-02   AG=4.58
 (BLK-CIRC=0) (#TAILS=1 OR MORE) (#CROSSHRKS=0) (#EXTREM=0 OR 1) (TEXTURE=BLANK,COARSE) (#EMP-CIRC=0 OR 1)
 (#EMP-TRIAN=0) (SHAPE=IRREGULAR,ELLIPSE,TRIANG-SQ) (#EYES=0)


(11)  MODE = FREE      DT = 0.000E+00      ST = 13      NGB =  6     RANK = 13     TYPE = DC     ULIST = 00000000001000
      CLIST =  6   1                TLIST =  0.000  0.000

THE FOLLOWING   1 CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES  10
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 2.067E-05 DENSITY THRESHOLD IS 0.000E+00
COMPLEX   1 OF RANK  5 COVERS    6 EVENTS (  6 NEW) WITH DENSITY OF 6.250E-02   AG=4.00
 (BLK-CIRC=2 OR MORE) (#TAILS=1 OR MORE) (#CROSSHRKS=3) (TEXTURE=BLANK) (#EMP-SQ=0) (TAIL=NONE,STRAIGHT)
 (SHAPE=IRREGULAR,TRIANG-SQ) (#EYES=0) (#BLK-SQ=0)
```

```
( 3)  MODE = FREE       DT = 0.000E+00      ST = 13      NGB =  6      RANK = 13      TYPE = DC      ULIST = 00100000000000
      CLIST =    6   1                    TLIST =      0.000  0.000

THE FOLLOWING   1  CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES   2
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 1.722E-05 DENSITY THRESHOLD IS 0.000E+00

COMPLEX   1 OF RANK  3 COVERS    5 EVENTS (  5 NEW) WITH DENSITY OF 4.167E-01  AG=1.26
 (BLK-CIRC=0)  (#TAILS=0)  (#CROSSHRKS=0)  (TEXTURE=BLANK)  (#EMP-CIRC=2,3 OR MORE)  (#EMP-SQ=0)  (#EMP-TRIAN=1 OR MORE)
 (TAIL=NONE)  (SHAPE=IRREGULAR,ELLIPSE,CIRCLE)  (#ANGLES=0)  (#EYES=0)  (#BLK-SQ=0)


( 4)  MODE = FREE       DT = 0.000E+00      ST = 13      NGB =  6      RANK = 13      TYPE = DC      ULIST = 00010000000000
      CLIST =    6   1                    TLIST =      0.000  0.000

THE FOLLOWING   1  CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES   3
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 1.722E-05 DENSITY THRESHOLD IS 0.000E+00

COMPLEX   1 OF RANK  4 COVERS    5 EVENTS (  5 NEW) WITH DENSITY OF 3.472E-02  AG=4.85
 (BLK-CIRC=0)  (#CROSSHRKS=0)  (TEXTURE=BLANK)  (#EMP-SQ=0)  (#EMP-TRIAN=0)  (TAIL=NONE)  (SHAPE=IRREGULAR,ELLIPSE,TRIANG-SQ)
 (#EYES=0)


( 5)  MODE = FREE       DT = 0.000E+00      ST = 13      NGB =  6      RANK = 13      TYPE = DC      ULIST = 00001000000000
      CLIST =    6   1                    TLIST =      0.000  0.000

THE FOLLOWING   1  CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES   4
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 2.067E-05 DENSITY THRESHOLD IS 0.000E+00

COMPLEX   1 OF RANK  2 COVERS    6 EVENTS (  6 NEW) WITH DENSITY OF 7.500E-01  AG=0.41
 (BLK-CIRC=0)  (#TAILS=1 OR MORE)  (#CROSSHRKS=0)  (TEXTURE=BLANK)  (#EMP-CIRC=0 OR 1)  (#EMP-SQ=0)  (#EMP-TRIAN=0)
 (TAIL=STRAIGHT)  (SHAPE=IRREGULAR,TRIANG-SQ)  (#EYES=0)  (#BLK-SQ=0)


( 6)  MODE = FREE       DT = 0.000E+00      ST = 13      NGB =  6      RANK = 13      TYPE = DC      ULIST = 00000100000000
      CLIST =    6   1                    TLIST =      0.000  0.000

THE FOLLOWING   1  CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES   5
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 2.067E-05 DENSITY THRESHOLD IS 0.000E+00

COMPLEX   1 OF RANK  4 COVERS    6 EVENTS (  6 NEW) WITH DENSITY OF 1.875E-01  AG=2.42
 (BLK-CIRC=0)  (#CROSSHRKS=0)  (#EXTREM=0 OR 1)  (TEXTURE=BLANK,VERT LINES)  (#EMP-CIRC=0 OR 1)  (#EMP-SQ=0)  (#EMP-TRIAN=0)
 (TAIL=NONE,STRAIGHT)  (SHAPE=IRREGULAR,TRIANG-SQ)  (#EYES=0)  (#BLK-SQ=0)


( 7)  MODE = FREE       DT = 0.000E+00      ST = 13      NGB =  6      RANK = 13      TYPE = DC      ULIST = 00000010000000
      CLIST =    6   1                    TLIST =      0.000  0.000

THE FOLLOWING   1  CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES   6
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 1.722E-05 DENSITY THRESHOLD IS 0.000E+00

COMPLEX   1 OF RANK  2 COVERS    5 EVENTS (  5 NEW) WITH DENSITY OF 1.250E+00  AG=-.32
 (BLK-CIRC=1)  (#TAILS=1 OR MORE)  (#CROSSHRKS=0)  (TEXTURE=BLANK)  (#EMP-CIRC=0 OR 1)  (#EMP-SQ=0)  (#EMP-TRIAN=0)
 (TAIL=STRAIGHT)  (SHAPE=IRREGULAR)  (#EYES=0)  (#BLK-SQ=0)
```

```
(15)  MODE = FREE      DT = 0.000E+00      ST =      NGB =  6     RANK =  3     TYPE = DC     ULIST = 11111111111111
      CLIST =  6  1     TLIST =  0.000  8  0.000

THE FOLLOWING   8  CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES   0   1   2   3   4   5   6   7   8   9  10
11  12  13
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 2.721E-04 DENSITY THRESHOLD IS 0.000E+00

COMPLEX   1 OF RANK  3 COVERS    47 EVENTS (  47 NEW) WITH DENSITY OF 4.080E-02   AG=4.62
 (BLK-CIRC=0) (#CROSSMARKS=0) (TEXTURE=BLANK) (#EMP-SQ=0) (#EYES=0)

COMPLEX   2 OF RANK  3 COVERS    20 EVENTS (  20 NEW) WITH DENSITY OF 1.736E-02   AG=5.85
 (BLK-CIRC=1..2 OR MORE) (#TAILS=1 OR MORE) (TEXTURE=BLANK) (TAIL=NONE,STRAIGHT) (SHAPE=IRREGULAR,TRIANG-SQ) (#EYES=0)
 (#BLK-SQ=0)

COMPLEX   3 OF RANK  3 COVERS     4 EVENTS (   4 NEW) WITH DENSITY OF 2.778E-02   AG=5.17
 (BLK-CIRC=0) (#TAILS=1 OR MORE) (#CROSSMARKS=0) (#EXTREM=0 OR 1) (TEXTURE=HORIZ LINE,COARSE,VERT LINES)
 (#EMP-CIRC=0 OR 1) (#EMP-SQ=0) (#EMP-TRIAN=0) (#EYES=0)

COMPLEX   4 OF RANK  1 COVERS     2 EVENTS (   2 NEW) WITH DENSITY OF 1.000E+00   AG=0.00
 (BLK-CIRC=0) (#TAILS=0) (#CROSSMARKS=0) (#EXTREM=2 OR MORE) (TEXTURE=BLANK) (#EMP-CIRC=0 OR 1) (#EMP-SQ=0)
 (#EMP-TRIAN=0) (TAIL=NONE) (SHAPE=IRREGULAR) (#EYES=1 OR MORE) (#BLK-SQ=0)

COMPLEX   5 OF RANK  3 COVERS     2 EVENTS (   2 NEW) WITH DENSITY OF 1.667E-01   AG=2.58
 (#TAILS=0) (#CROSSMARKS=0) (#EXTREM=2 OR MORE) (TEXTURE=WAVES,DIAG LINES) (#EMP-CIRC=0 OR 1) (#EMP-SQ=0) (#EMP-TRIAN=0)
 (TAIL=NONE) (SHAPE=IRREGULAR) (#EYES=0) (#BLK-SQ=0)

COMPLEX   6 OF RANK  1 COVERS     2 EVENTS (   2 NEW) WITH DENSITY OF 1.000E+00   AG=0.00
 (BLK-CIRC=0) (#TAILS=1 OR MORE) (#CROSSMARKS=0) (#EXTREM=0 OR 1) (TEXTURE=BLANK) (#EMP-CIRC=0 OR 1) (#EMP-SQ=1 OR MORE)
 (#EMP-TRIAN=0) (TAIL=SPRING) (SHAPE=IRREGULAR) (#EYES=0) (#BLK-SQ=1 OR MORE)

COMPLEX   7 OF RANK  0 COVERS     1 EVENTS (   1 NEW) WITH DENSITY OF 1.000E+00   AG=0.00
 (BLK-CIRC=1) (#TAILS=0) (#CROSSMARKS=0) (#EXTREM=0 OR 1) (TEXTURE=BLANK) (#EMP-CIRC=0 OR 1) (#EMP-SQ=0) (#EMP-TRIAN=0)
 (TAIL=NONE) (SHAPE=TRIANG-SQ) (#ANGLES=1 OR MORE) (#EYES=0) (#BLK-SQ=0)

COMPLEX   8 OF RANK  0 COVERS     1 EVENTS (   1 NEW) WITH DENSITY OF 1.000E+00   AG=0.00
 (BLK-CIRC=0) (#TAILS=0) (#CROSSMARKS=0) (#EXTREM=2 OR MORE) (TEXTURE=BLANK) (#EMP-CIRC=0 OR 1) (#EMP-SQ=1 OR MORE)
 (#EMP-TRIAN=0) (TAIL=NONE) (SHAPE=IRREGULAR) (#ANGLES=0) (#EYES=0) (#BLK-SQ=0)


THE FOLLOWING SELECTORS ARE COMMON CHARACTERISTICS
 ( NONE )
```

```
(12)  NODE = FREE        DT = 0.000E+00      ST = 13      NGB =  6      RANK = 13      TYPE = DC      ULIST = 00000000000100
      CLIST =     6   1          TLIST =     0.000   0.000

THE FOLLOWING   1 CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES  11
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 2.411E-05 DENSITY THRESHOLD IS 0.000E+00

COMPLEX   1 OF RANK  4 COVERS    7 EVENTS (  7 NEW) WITH DENSITY OF 4.861E-02    AG=4.36
 (BLK-CIRC=1..2 OR MORE) (#TAILS=1 OR MORE) (#EXTREM=0 OR 1) (TEXTURE=BLANK) (#EMP-TRIAN=0) (TAIL=NONE,STRAIGHT)
 (SHAPE=IRREGULAR) (#EYES=0) (#BLK-SQ=0)

(13)  NODE = FREE        DT = 0.000E+00      ST = 13      NGB =  6      RANK = 13      TYPE = DC      ULIST = 00000000000010
      CLIST =     6   1          TLIST =     0.000   0.000

THE FOLLOWING   1 CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES  12
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 1.722E-05 DENSITY THRESHOLD IS 0.000E+00

COMPLEX   1 OF RANK  3 COVERS    5 EVENTS (  5 NEW) WITH DENSITY OF 3.125E-01    AG=1.68
 (BLK-CIRC=0..1) (#TAILS=0) (#CROSSBRKS=0) (#EXTREM=2 OR MORE) (TEXTURE=BLANK,DIAG LINES) (#EMP-CIRC=0 OR 1)
 (TAIL=NONE) (SHAPE=IRREGULAR) (#ANGLES=0) (#EYES=0) (#BLK-SQ=0)

(14)  NODE = FREE        DT = 0.000E+00      ST = 13      NGB =  6      RANK = 13      TYPE = DC      ULIST = 00000000000001
      CLIST =     6   1          TLIST =     0.000   0.000

THE FOLLOWING   1 CARTESIAN COMPLEXES FORM THE UNICLASS COVER FOR CLASSES  13
     DENSITY OF LEARNING EVENTS IN EVENT SPACE IS 2.067E-05 DENSITY THRESHOLD IS 0.000E+00

COMPLEX   1 OF RANK  2 COVERS    6 EVENTS (  6 NEW) WITH DENSITY OF 7.500E-01    AG=0.41
 (BLK-CIRC=0) (#TAILS=0) (#CROSSBRKS=0) (#EXTREM=2 OR MORE) (TEXTURE=BLANK,WAVES) (#EMP-CIRC=0 OR 1) (#EMP-SQ=0)
 (#EMP-TRIAN=0) (TAIL=NONE) (SHAPE=IRREGULAR) (#BLK-SQ=0)
```