AQLISP: A LISP Program for Inductive Generalization of VL1 Event Sets

P. Richards

MLI 79-9

A publication of the *Machine Learning and Inference Laboratory*
Artificial Intelligence Center
George Mason University
Fairfax, Virginia 22030-4444 U.S.A.
(703) 764-6259

Editor: R. S. Michalski
Assistant Editors: J. Ralston and J. Zhang

The *MLI Reports* replace *ISG Reports*
published until December 1987
by the Artificial Intelligence Laboratory,
Department of Computer Science
University of Illinois,
Urbana, Illinois 61801

77-9

AQLISP:
A LISP Program for Inductive Generalization of VL_1 Event Sets

Paul Richards

Internal Report 12-15-79
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

AQLISP:
A LISP Program for Inductive Generalization of VL_1 Event Sets

This paper describes the operation and internal structure of a program called AQLISP, revision 10-Apr-79. AQLISP is an interactive LISP 1.5 program for generalization and optimization of discriminant descriptions of object classes. The descriptions are expressed as disjunctive normal expressions in variable valued logic system VL_1 [6]. Such expressions are unions of conjunctive statements (complexes) involving relations on multiple-valued variables. Input data to the program are sets of VL_1 events (sequences of attribute-value pairs) describing individual objects. Each set of events is associated with a given class name.

It is beyond the scope of this paper to describe the VL1 system, or algorithm AQ in detail. The reader is referred to [6] for further information on VL1, and [7,8] for a detailed description of the theory behind the algorithm implemented.

1 Why AQ in LISP?

There are several motivations for implementing algorithm AQ in LISP. The most significant one is the ease with which dynamic structures are constructed and destroyed, which is essential to the algorithm. Additionally, LISP lends itself to interactive communication with the user. This allows the program to be used as a "sounding board" for testing new ideas quickly and easily. Finally, many of the applications for AQ are in areas of pattern recognition and artificial intelligence, which are often explored using LISP programs.

Section 2 describes the operation of the top level function (AQVAL),

to interactively use the AQLISP system. Described are the formats of event and class input, parameter specification, mode selection, and output format. Sections 2.7 and 2.8 provide an example of the interactive system dialog and the output generated by the various modes. Section 3 describes the data structures required by the system, and a description of the principal interface to the implementation, function (AQ), which permits use of the AQ system within user written programs. Section 4 offers some comparisons of AQLISP and AQ7, a PL/1 implementation of algorithm AQ. Listings of the system functions, and installation dependent operating instructions are provided in the appendices.

2 Interactive Operation.

Once the AQLISP system is loaded into a LISP interpreter, it may be invoked interactively by calling the function "(AQVAL)". AQVAL then begins an interactive dialog with the user, requesting (1) parametric information, (2) variable descriptions, (3) event and class specifications, and (4) an operating mode. AQLISP uses the standard LISP functions (READ), (PRIN1 ...), (PRINT ...), and (TERPRI) for all input and output, and assumes that these are directed to the users terminal. It also assumes (READ) echos input provided to it; AQLISP does no echoing of its input. (READ) must also allow several S-expressions to be supplied by the user on the same physical line. The dialog of each section mentioned above is demonstrated by showing the prompt provided by AQLISP on the left, and an

explanation of possible answers to the right. (See Appendix A for installation-specific properties of the interactive system).

2.1 Parametric Specifications.

Currently, two parameters may be specified interactively to the AQLISP system: MAXSTAR, the maximum star size permitted when generating stars, and CUTSTAR, the size to which a star will be trimmed when it exceeds MAXSTAR in size. These questions run as follows:

ENTER MAXIMUM STAR SIZE FOR THIS RUN:

Enter an integer number greater than zero. The upper bound on this parameter is determined by space available to the program, and CPU time available for processing. Too large a value may cause excessive garbage collections, complete exhaustion of free-space, or use very large amounts of computer time. Too small a value will cause excessive trimming of the stars generated, which lowers the probablilty that a truly optimal cover will be found. (Note that if any trimming is done at all, the covers generated by AQ are not necessarly optimal).

ENTER SIZE TO CUT STAR WHEN TRUNCATING:

Enter an integer number greater than zero, and less than CUTSTAR, mentioned above. The closer this value is to CUTSTAR, overhead caused by star trimming will increase, with subsequent increase in CPU time used.

The parameter that determines optimality criteria, CRITERIA&TOLERANCELIST,

is always initialized to ((#COVERED 0.0) (NUMBEROFSELECTORS 0.0)), which causes the covers produced by the interactive system to always be selected from the complexes that cover the largest number of events, and in the case of ties, those with the fewest number of terms in VL1 format. The parameter RESTRICTIONLIST is always initialized to NIL. (See section 3.3 for more information on the parameters.)

2.2 Defining Variables.

Every variable to be used in the current AQLISP run must be explicitly declared after entering parameters. Additionally, the type and domain of these variables must be provided. Variables may either be NOMINAL (assume discrete values) or LINEAR (assume interval values on the range [0 .. <maximum integer on system>]).

DEFINE VARIABLES FOR YOUR PROBLEM:

(END WITH '*')

This message indicates variable declarations are to be entered. Variables should be entered in groups that have identical domains and types. After all groups are entered and the program again prompts for variable names, enter '*' to indicate you wish to enter classes and events.

VARIABLE NAME(S):

Enter names of all variables that belong to the next group of identical domains and

types. Variable names should be composed of the letters A-Z and digits 0-9, and should start with a letter (conforming to the installation conventions for non-numeric atoms - see appendix A). Several variables may be entered on a line (or several lines), seperated by either commas or spaces. The last variable should be followed by the symbol '*' to denote the end of the name list.

If all variables have been entered, enter only '*' to start entry of events and classes.

DOMAIN TYPE:

Two possible answers may be specified here. Either enter:

NOMINAL
: to specify that these variables may assume only discrete values that are elements of the domain specified in the next question, or

LINEAR
: to specify that these variables assume interval values on the range of non-negative integers [0 .. <max integer on system>]. The domain of these variables is also specified in the next question.

If the domain type NOMINAL was specified, the following prompt is issued:

WHAT ARE THE PERMISSIBLE VALUES:
(GIVE VALUES, SEPERATED BY SPACES OR COMMAS, END WITH '*')

Enter all the possible values the current variable(s) may assume. These may be any atomic symbol permitted by the LISP implementation. (Note - a restriction on the total number of nominal values is imposed - see secton 3 and Appendix A). The last value specified should be followed by a space or comma, and a '*'.

If the domain type specified was LINEAR, the following prompt is issued:

```
WHAT IS THE RANGE OF PERMISSIBLE VALUES:
(TYPE AS 'LOWVALUE,HIGHVALUE')
```

Enter the range of values or intervals the current variables may contain. The limits should be elements of the integer interval [0 .. maximum integer on system>], with LOWVALUE<HIGHVALUE. Only one interval may be specified for the domain of a linear variable.

After the domain of the current group of variables is specified, AQLISP will prompt again for variable names. If more variables are to be defined, enter another group. If there are no more variables to be defined, enter '*', and the system will begin requesting events and classes.

2.3 Entering Events and Classes.

Events are entered to AQLISP in a VL1-like format, one VL1 formula to each event specified. The VL1 formulae are formed by the conjunction of terms (or selectors). Each selector takes either of two forms:

1) for a NOMINAL variable, the selector format is

[<var> <relation> <val_1> ... <val_n>]

where

<var> is a NOMINAL type variable

<relation> is either "=" or "#" to specify equality or inequality to the values specified

<val_i> is a valid value from this variable's domain. More than one value may be specified to indicate several values belong to the same event.

2) for a LINEAR variable, the selector format is:

[<var> <relation> <lowvalue> .. <highvalue>]
or
[<var> <relation> <value>]

where

<var> is a LINEAR variable

<relation> is either "=" or "#"

<lowvalue> is the lower bounding value of the interval for the event

<highvalue> is the upper bounding value of the interval for the event

<value> may be specified if only a specific point value is to be included, i.e., when <lowvalue>=<highvalue>.

Spaces must be placed between the variable name, relation, and values, as these must be input to AQLISP as seperate atoms. The conjunction of selectors (i.e. complexes) is indicated by entering the selectors adjacent to each other, and the end of the conjunctive formula is indicated by a ";". Thus, if L1 and L2 are linear variables on [0 .. 4], and N1 is a nominal variable with domain (A, B, C), a complete VL1 expression for an event could be:

[L1 = 2 .. 3][N1 = B][L2 = 1];

The square brackets around the terms may be replaced by parenthesis, if desired or if they do not represent super-parenthesis on the current LISP.

Class names can be any atomic symbol the user desires. The dialog for class and event entry runs as follows:

ENTER EVENTS AND CLASSES:

EVENT:

A VL1 formula expressing a new event (complex) should be entered here. If no more events are to be specified, enter a ";".

The VL1 formula may be continued over more than one line if the (READ) function permits continuation of S-expressions.

CLASS:

The class name to which the preceeding event belongs should be typed here. Any atomic symbol is permitted as a class name.

2.4 Operating mode selection.

There are four "modes" in which the AQ algorithm may be applied to the event clusters: (1) Intersecting covers, (2) Non-intersecting covers, (3) Disjointcomplex covers, and (4) Sequential covers.

2.4.1 Intersecting covers -

Intersecting covers are generated by applying AQ in the following manner: let E_i represent the set of events to be covered for class i and F be the set of all events specified to the system. Each cover C_i is constructed by applying AQ to E_i against $F - E_i$. Thus, the intersection of

any two covers $C_i \cap C_j$, $i \neq j$ may be non-null. The intersection will not contain any event points originally specified as an event, it only can occur over unspecified events.

2.4.2 Non-intersecting covers -

Nonintersecting covers are generated by applying AQ in this manner: Sort the classes into alphabetic order by classname, and assign each class a unique index i. Thus, E_1 = <events in class with 'first' alphabetic name>, and so on. Let $F = \underset{all}{U}\{E_i\}$, i.e., all events explicitly specified. The covers for class i are generated by applying AQ to E_i against $F - E_i + U\{C_j\}$, where C_j = <cover of class j>. In this manner, it is guaranteed that $C_i \cap C_j = \{\}$, $i \neq j$.

2.4.3 Disjoint Complexes -

Disjoint complex covers are produced in a similar manner to Non-intersecting covers, except that the star generation of AQ is performed in such a way to guarantee each complex in the cover is disjoint. In the Intersecting and Non-intersecting cover modes, this may not be the case - complexes within the same cover can have non-null intersections in those modes.

2.4.4 Sequential mode -

Sequential mode produces covers in the following manner: first, the classes are sorted as before in Non-intersecting covers. Then, the cover Ci for class i is produced by applying AQ to E_i against F - E_j. To utilize the covers produced by this mode, they must be tested in the same sequential order that they were constructed (ie, alphabetic classname order). Each cover may contain any event points allocated to a previously generated cover. This mode is useful in classifying new events with a minimum of variable testing, since fewer complexes are needed to specify some covers.

2.4.5 Specifying the operating mode.

The operating mode choice is entered in response to:

MODE OF OPERATION IS:

enter:
INTERSECTINGCOVERS,
NONINTERSECTINGCOVERS,
DISJOINTCOMPLEXES, or
SEQUENTIAL
to select the mode desired.

After the mode is selected, AQ will be applied as described, and the covers will be computed and printed. After all of the covers are printed the following question is printed:

DO YOU WANT TO TRY ANOTHER MODE?

Enter YES if you want to try another mode on the same data, otherwise enter NO. This allows you to try any of the four modes on the same data. If YES is entered, the MODE ... prompt is re-issued. If NO is entered, a garbage collection is performed, and (AQVAL) returns to the calling function (usually top-level EVAL).

2.5 Program Output.

Interactive AQLISP produces its covers as a series of disjunctive VL1 formulas for each class. Each is printed in the same VL1 format as was used for input. The output appears as:

```
THE COVERS ARE:

COVER OF CLASS: <first class in alpha order>
[<vl term 1>][<vl term 2>]...[<vl term n>]
[<vl term n+1>][<vl term n+2>]...[<vl term n+m>]
 .
 .
 .
```

where each line represents a seperate interval. The cover is the union of all intervals printed under the heading line. This is repeated for each class AQ covered. If no intervals are printed, the cover is the entire event space.

2.6 Other messages printed by AQLISP

Certain questions accept only a few specific answers, such as when

mode is to be selected, or the domain type of a variable is defined. If the message:

```
Not an acceptable answer!
Please enter one of these:
     <answer choice 1>
     <answer choice 2>
          .
          .
```

is printed, the user has entered a response that is not acceptable. One of the choices indicated must be entered. On terminals equipped with both upper and lower case, case shifting may be significant.

Several places within AQLISP, consistancy checks are made on the internal structures of AQ. If unexpected forms are detected, an undefined function (SHOULDNT) is invoked to force LISP to abort the run. These aborts are usually caused by improper data entry of variables, events, or class names (mismatched parenthesis, omitted spaces, non-atomic names, etc.) After one of these aborts, AQVAL must be started from the beginning dialog.

2.7 Sample AQLISP Dialog.

Below is an example terminal dialog, entering variables, events/classes, and associated output that will be used in section 2.8. (User-entered text is underlined).

```
WELCOME TO AQLISP: REVISION 10-APR-79


ENTER MAXIMUM STAR SIZE FOR THIS RUN:
25

ENTER SIZE TO CUT STAR TO WHEN TRUNCATING:
10

DEFINE VARIABLES FOR YOUR PROBLEM: (END WITH '*')
VARIABLE NAME(S):
V1 V3 *
DOMAIN TYPE:
NOMINAL
WHAT ARE THE PERMISSIBLE VALUES:
(GIVE VALUES, SEPARATED BY SPACES OR COMMAS, END WITH '*')
1 2 *

VARIABLE NAME(S):
V2 V4 *
DOMAIN TYPE:
NOMINAL
WHAT ARE THE PERMISSIBLE VALUES:
(GIVE VALUES, SEPARATED BY SPACES OR COMMAS, END WITH '*')
1 2 3 *

VARIABLE NAME(S):
*


ENTER EVENTS AND CLASSES:

EVENT:
[V1 = 1][V2 = 1][V3 = 1][V4 = 1];
CLASS:
I

EVENT:
[V1 = 1][V2 = 3][V3 = 1][V4 = 3];
CLASS:
I

EVENT:
[V1 = 2][V2 = 2][V3 = 2][V4 = 3];
CLASS:
I
```

```
EVENT:
[V1 = 1] [V2 = 2] [V3 = 1] [V4 = 2];
CLASS:
II

EVENT:
[V1 = 1] [V2 = 2] [V3 = 2] [V4 = 1];
CLASS:
II

EVENT:
[V1 = 1] [V2 = 1] [V3 = 2] [V4 = 2];
CLASS:
II

EVENT:
[V1 = 2] [V2 = 1] [V3 = 1] [V4 = 2];
CLASS:
III

EVENT:
[V1 = 2] [V2 = 3] [V3 = 1] [V4 = 2];
CLASS:
III

EVENT:
[V1 = 2] [V2 = 3] [V3 = 2] [V4 = 2];
CLASS:
III

EVENT:
;

MODE OF OPERATION IS:
INTERSECTINGCOVERS

THE COVERS ARE:
COVER OF CLASS:  I
[V4 = 3]
[V2 = 1,3] [V4 = 1,3]

COVER OF CLASS:  II
[V1 = 1] [V3 = 2]
[V1 = 1] [V2 = 2]
```

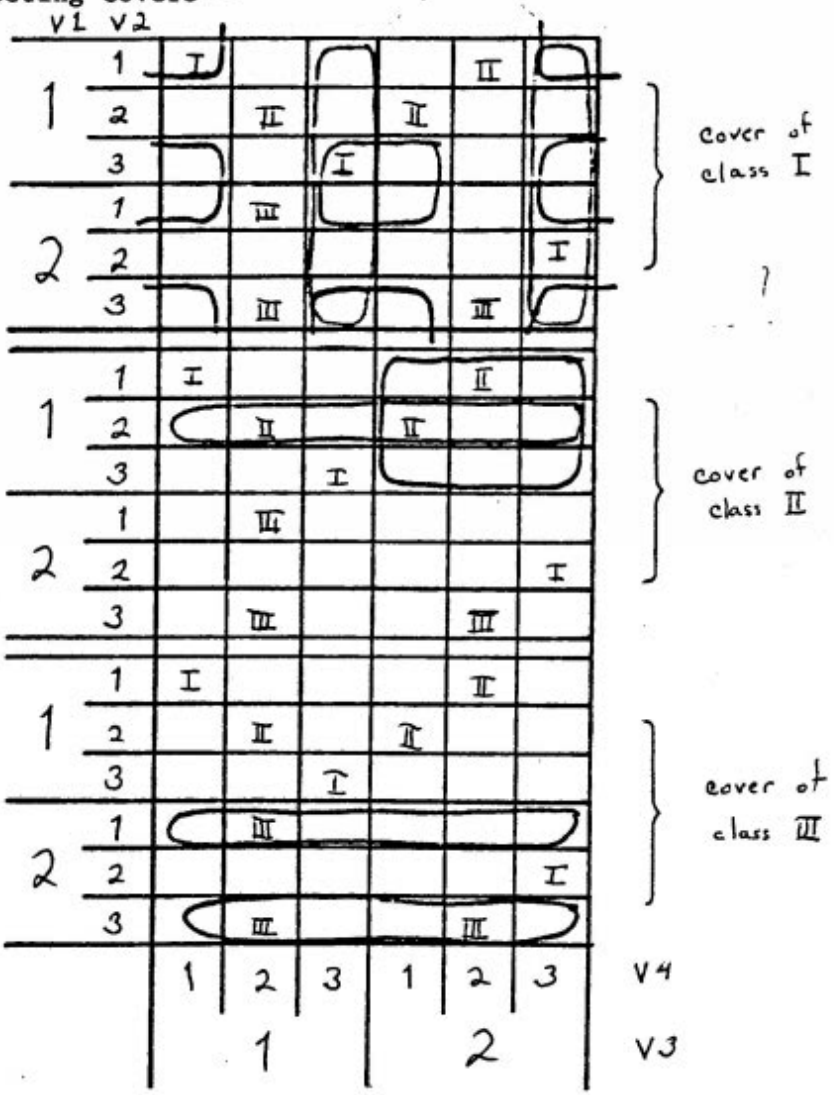
```
COVER OF CLASS:  III
[V1 = 2] [V2 = 1,3]

DO YOU WANT TO TRY ANOTHER MODE?
YES

MODE OF OPERATION IS:
NONINTERSECTINGCOVERS

THE COVERS ARE:
COVER OF CLASS:  I
[V4 = 3]
[V2 = 1,3] [V4 = 1,3]

COVER OF CLASS:  II
[V1 = 1] [V4 = 2]
[V2 = 2] [V4 = 1,2]

COVER OF CLASS:  III
[V1 = 2] [V2 = 1,3] [V4 = 2]

DO YOU WANT TO TRY ANOTHER MODE?
YES

MODE OF OPERATION IS:
DISJOINTCOMPLEXES

THE COVERS ARE:
COVER OF CLASS:  I
[V1 = 2] [V2 = 2]
[V2 = 1,3] [V4 = 1,3]

COVER OF CLASS:  II
[V1 = 1] [V2 = 1,3] [V4 = 2]
[V1 = 1] [V2 = 2]
```

```
COVER OF CLASS:  III
[V1 = 2] [V2 = 1,3] [V4 = 2]


DO YOU WANT TO TRY ANOTHER MODE?
YES


MODE OF OPERATION IS:
SEQUENTIAL


THE COVERS ARE:
COVER OF CLASS:  I
[V4 = 3]
[V2 = 1,3] [V4 = 1,3]


COVER OF CLASS:  II
[V1 = 1]


COVER OF CLASS:  III


DO YOU WANT TO TRY ANOTHER MODE?
NO

THIS RUN USED 12.954 SECONDS CPU TIME
```

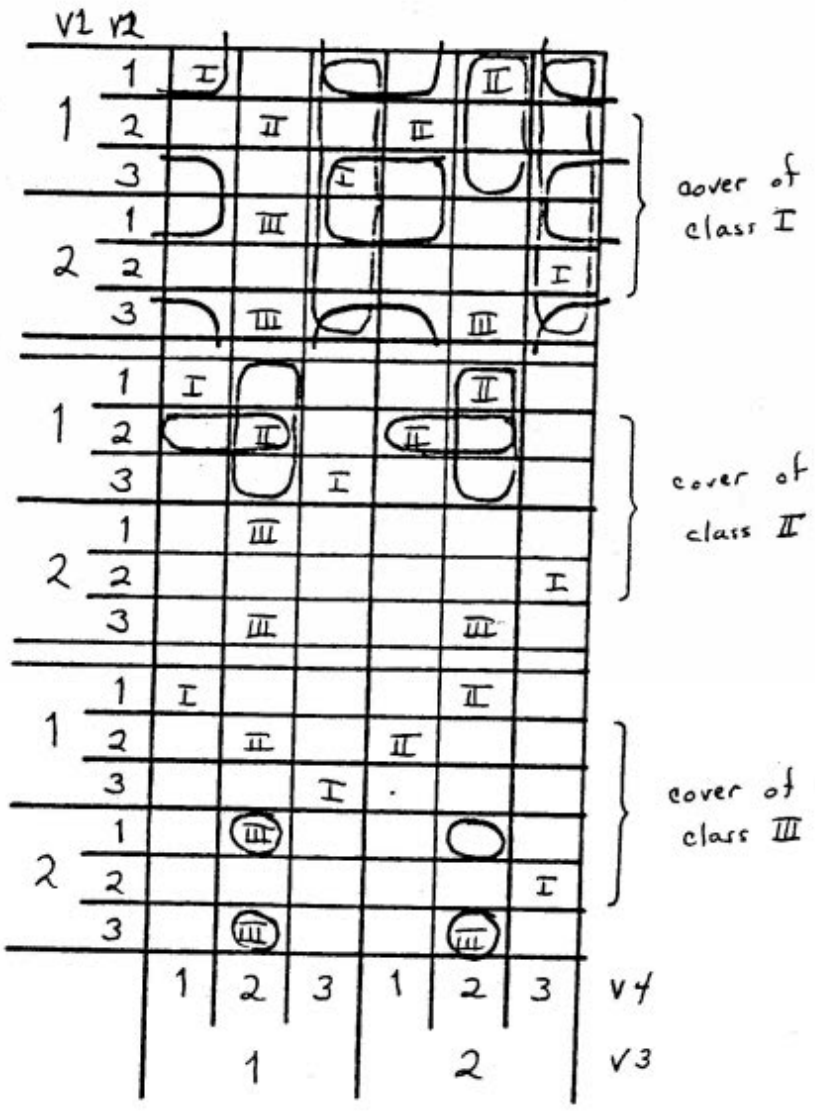
2.8 Examples of different operating modes.

Below are displayed (using "generalized logical diagrams") the covers produced by the preceeding dialog on the interactive system.

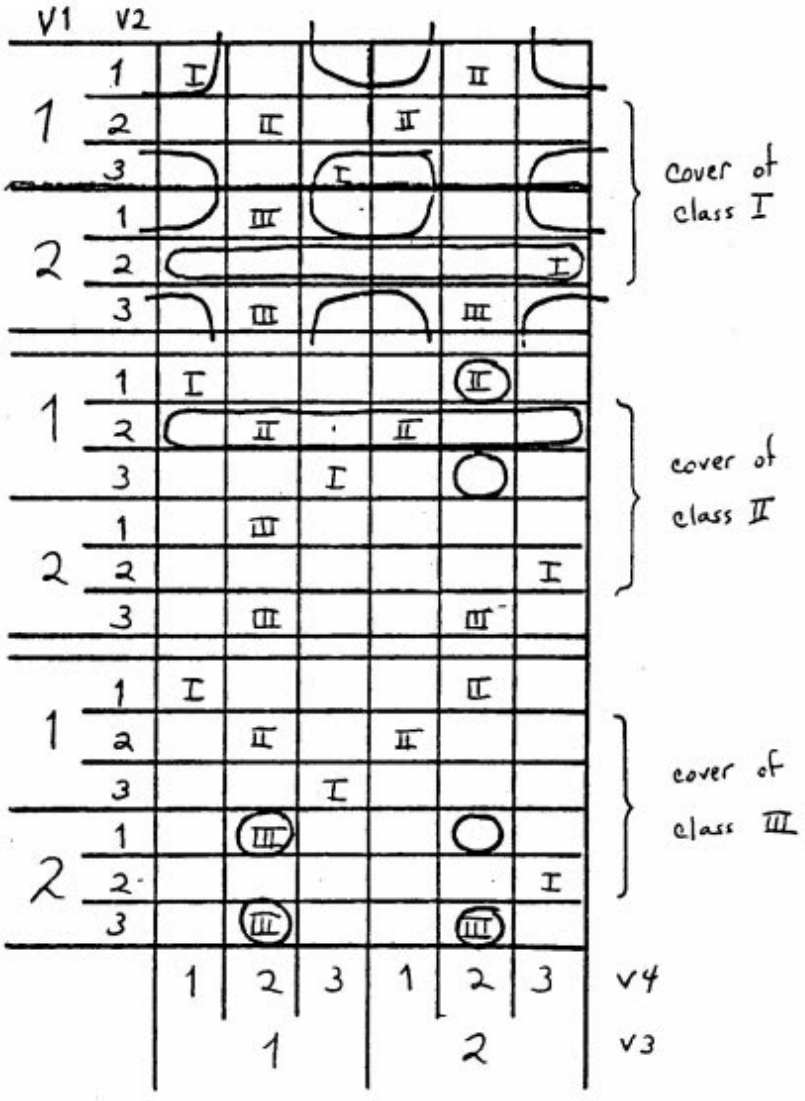
2.8.1 Intersecting covers -



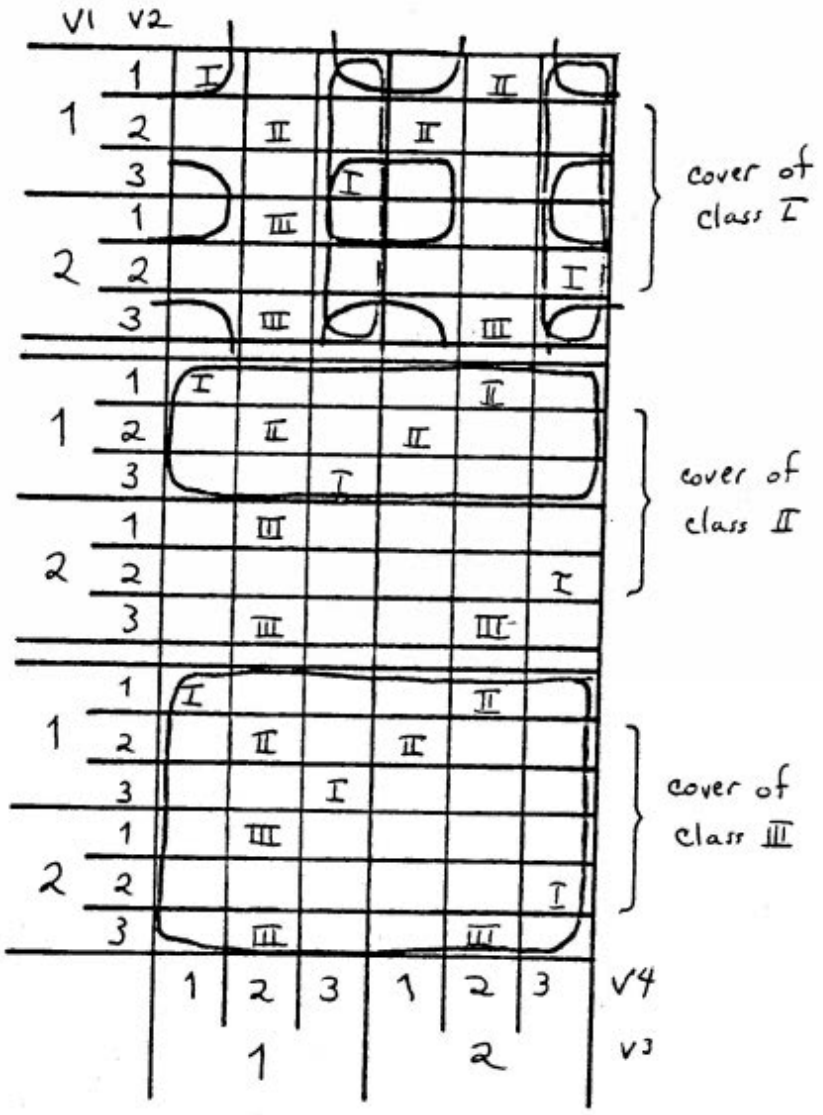
2.8.2 Nonintersecting covers -



2.8.3 Disjointcomplexes -



2.8.4 Sequential -



3 Using the AQ function internally

In addition to using the AQVAL driver for the AQ algorithm, it is possible to use the AQ function within other LISP programs, provided that the proper data structures are constructed prior to calling AQ. The next few sections deal with the construction of these structures, and the actual invocation of AQ.

3.1 Variable definitions.

The most basic structure in AQ is the variable. These are represented by non-numeric atoms with several special properties on their property lists. All variables referenced in any events must be prepared prior to calling AQ. We will divide variables into two different types: NOMINAL and LINEAR.

3.1.1 Nominal variables

Nominal variables are those that can only assume distinct discrete values. Each value permitted to a nominal variable is represented by a bit position in a full-word numeric atom. Bit positions are assigned starting with the least significant bit and moving to the most significant position. All values of all nominal variables must fit into one

full-word. See appendix A for installation size limits on full-word spaces.

The atoms that are declared to be NOMINAL must appear in a list named "NOMINALVARS". The order that variables appear in this list must correspond to the order that bit groups are assigned to the value set. The property "VALUESET" on each variable represents a list of possible values (literal atoms) that the variable may assume, the values also in the order of the assignments to bits within a bit group. Property "MASK-BIT" represents a numeric atom that contains a single bit aligned with the first bit position allocated to this variable in the nominal full-word. Property "MASK" contains a numeric atom consisting of all possible bits in this variables group (all posible values). These bits correspond to the values in the VALUESET list, and must be adjacent. For example, if N1, N2, and N3 are nominal variables with:

```
NOMINALVARS = (N1 N3 N2)

valueset(N1) = (A B C)
valueset(N2) = (D E)
valueset(N3) = (A F)
```

then

```
MASKBIT of N1 = 1Q (Q suffix indicates octal)
MASK    of N1 = 7Q

MASKBIT of N2 = 40Q
MASK    of N2 = 140Q

MASKBIT of N3 = 10Q
MASK    of N3 = 30Q
```

where

bit 0 (1Q) represents value	A of N1
bit 1 (2Q)	B of N1
bit 2 (4Q)	C of N1
bit 3 (10Q)	A of N3 (different from A of N1)
bit 4 (20Q)	F of N3
bit 5 (40Q)	D of N2
bit 6 (100Q)	E of N2

Bits are set to indicate the presence of a value. In addition, global list MASKLIST must be a list of all MASKs corresponding to variables in NOMINALVARS. An internal function, (SETUPMASKLIST NOMINALVARS), is provided to make these bit assignments for the user if desired. Finally, each NOMINAL variable must have property DOMAINTYPE with value "NOMINAL".

3.1.2 Linear variables

Linear variables are those that can assume either point values or interval values on the non-negative integers. Their declarative structure is quite simple: each linear variable must be a member of list LINEARVARS, and must have property DOMAINTYPE set to "LINEAR". Also, three additional properties must be constructed on each LINEAR variable:

LOW - lowest integer value this variable may assume. Must be non-negative.

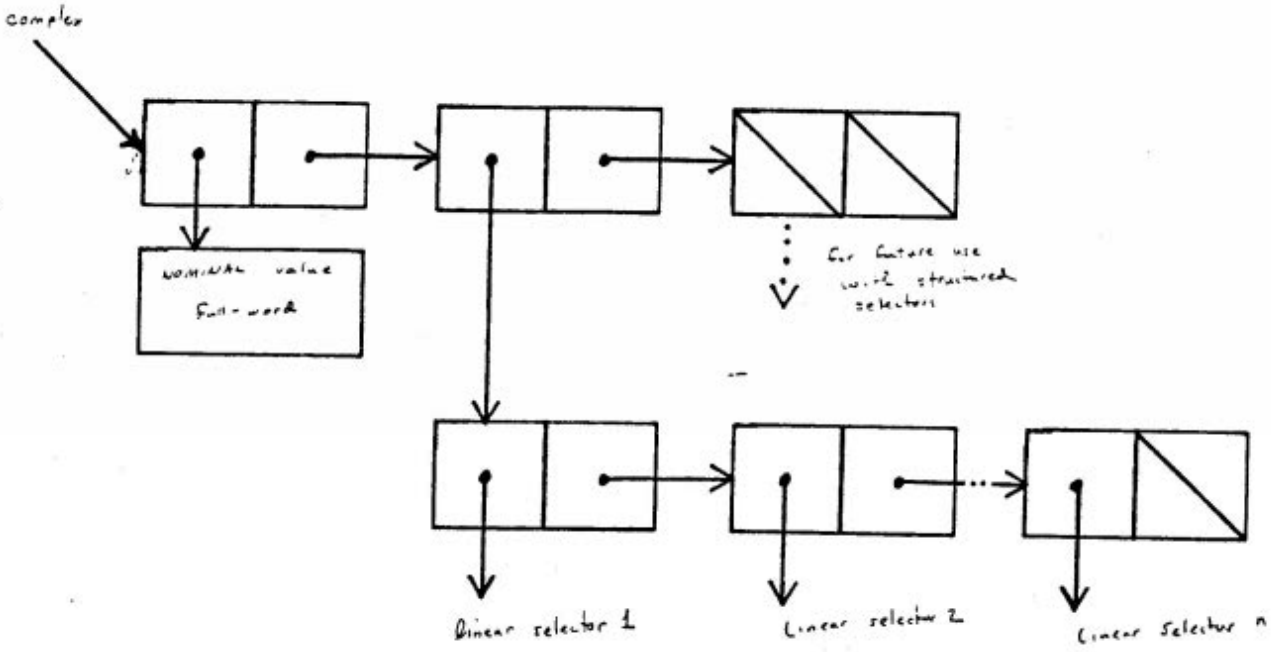
HIGH and
MAXBOUND - highest integer value this variable may assume. Must be non-negative, and greater than or equal to LOW(var)

Note that these structures only define the domain of each variable, and do

not assign values to the variable.

3.2 Event and Class structures.

Events in AQLISP are specified by the conjunction of VL1 selectors. Selectors specify the specific values variables hold at particular event points. This conjunction of selectors forms a COMPLEX. A complex in AQLISP is represented by the following structure:



Note all nominal selectors are collapsed into one full-word, but an arbi-

trary number of linear selectors may be specified for an event, limited only by free space. The list of linear selectors in sorted into order using function (EARLIER sel_1 sel_2) to indicate the ordering. This structure is built during interactive use by (READEVENTS). Classes are represented by lists of these complexes. In the interactive version, two global lists are maintained - "CLASSLIST", which is a list of all classes, and "CLASSNAMES", a list of all names associated with the corresponding item in CLASSLIST.

3.3 Parameter specifications

Several parameters must be supplied to AQ by global lists or variables. These are:

MAXSTAR
: set to a non-negative integer value that specifies the maximum size a star may assume during star generation in AQ.

CUTSTAR
: set to a non-negative integer number less than MAXSTAR that specifies the size to which a star is trimmed when its size exceeds MAXSTAR.

CRITERIA&TOLERANCELIST
: a list of doublets in the following format: (<function-name> <tolerance>), where <tolerance> is between 0.0 and 1.0. These are used to compute the cost factors used to trim complexes from a star when necessary. When called, the function is supplied with three arguments:

    1) a complex
    2) a list of complexes from all 'other' classes (those events not to be covered)

3) all complexes in the class to be covered that have not yet been covered.

Using these arguments, the function should compute and return a numeric "cost" representing the expense of including the specified complex in the star. The numeric values of the cost can be of an arbitrary scale, as long as increasing cost is represented by a more positive number returned. The <tolerance> is used to compute a allowable range of optimality. This range is computed by first applying the function to all complexes in the star, to determine the maximum and minimum cost of elements in the star. A limit is then computed as

LIM = min + (max - min)*<tolerance>

If a complex's cost is less than or equal to LIM, it is considered optimal, and is not trimmed unless only optimal intervals remain in the star, and the star must still be trimmed further.

These cost functions are applied in the order in which they appear in the CRITERIA&TOLERANCE list. This determines the most significant criteria, and the successive tie-breaking criteria.

Two cost functions are provided in the AQLISP source - #COVERED which computes how many events in the yet-to-be-covered list are covered by the complex, and NUMBEROFSELECTORS, which computes the cost of the number of selectors used to represent the complex.

RESTRICTIONLIST

must be set to NIL. This parameter is being used for future development in AQLISP.

3.4 The AQ function.

Once all variables are defined, and all events are constructed and organized into classes, AQ may be invoked by the following LISP call:

(AQ ELIST FLIST DISJOINTCOMPLEXFLG)

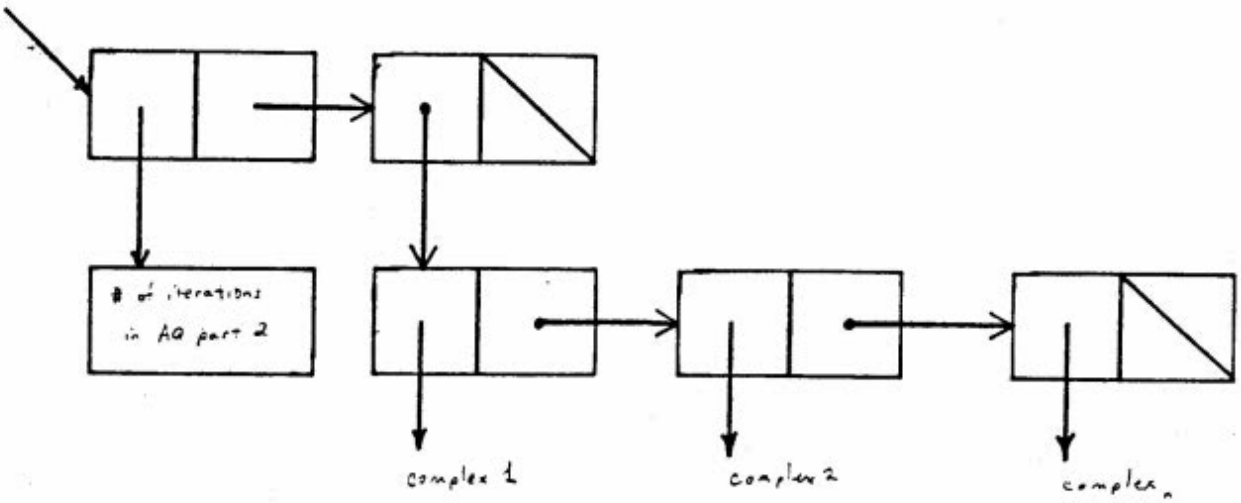
where

ELIST is a list of all complexes in the class to be covered by AQ

FLIST is a list of all other specified complexes that are <u>not</u> to be covered by AQ.

DISJOINTCOMPLEXFLG is NIL if the cover can be constructed of intersecting complexes, or T if the cover must be constructed of disjoint complexes.

(AQ) returns the following structure:



The union of events covered by all complexes is the cover of the specified class ELIST. The complxes may be decoded by reversing the process used to construct complexes - see function "PPCOMP" in Appendix B for an example of this.

3.5 Other functions available.

Many internal functions may prove useful to the user utilizing AQ as a callable function. Appendix B lists the entire AQLISP system, with brief comments about each function. The functions are grouped into four catagories - those used directly to implement algorithm AQ, data structure manipulation functions, input/output functions, and library support functions.

4 Comparison with other implementations of AQ

AQLISP is but one of several approaches to implementing algorithm AQ. Another readily accessible implementation is AQ7, a PL/1 based system. There are several important differences between AQLISP and AQ7, which will be described below.

Internally, the method of storing the events, complexes, and variables are radically different between AQ7 and AQLISP. AQ7 stores all event-related information as sequences of bit strings, with the obvious computational efficiency benifiting this approach. However, this imposes some size constraints on problem domains that AQ7 can process (<64 events, <64 variables). Also, AQ7 has an advantage of being in a compiled language, which dramatically improves execution time. AQ7 is currently available only in a batch environment.

AQLISP on the other hand, offers greater flexibility than AQ7, at the expense of execution efficiency. Test runs on small (<10 variables) problems show an increase of at least a factor of 10 in execution time over AQ7, even when AQLISP is run on a CYBER 175, versus AQ7 on a IBM 360/75.

However, AQLISP offers an interactive interface, which makes it convienent to use when testing new ideas. Also, AQLISP is not as restricted on the size of the problem domain as AQ7, since an unlimited number of linear selectors with much greater range than AQ7 can be accomodated (see 3.3). In addition, AQLISP has potential to handle other selector formats than interval and nominal.

5 Conclusions

The operating characteristics of AQLISP has been described and demonstrated. An interactive interface to the AQ function has been documented, and a sample execution shown. Finally, the calling requirements of AQ are illustrated, and the primary data structures diagrammed. A complete program listing of AQ appears in Appendix B.

References

1. -- , Interlisp Reference Manual, Warren Teitelman, ed., Xerox Palo Alto Research Center, California, 1974.

2. Bobrow, D.G., and R. S. Michalski, Source listings for AQLISP on Interlisp-10, Installed on Bolt, Beranek, and Newman system BBND, 1978.

3. Greenwalt, E. M., Johnathan Slocum, and Robert A. Amsler, UT LISP Documentation, Version 4.0, University of Texas at Austin Computation Center, 1975.

4. Larson, James, and R. S. Michalski, "AQVAL/1 (AQ7) Users Guide and Program Description", Report number 731, Department of Computer Science, University of Illinois, Urbana, June 1975.

5. Michalski, R. S., "A Geometrical Model for the Synthesis of Interval Covers", Report number 461, Department of Computer Science, University of Illinois, Urbana, June, 1971.

6. Michalski, R. S., "Variable-Valued Logic: System VL1,", 1974 International Symposium on Multiple-Valued Logic, West Virginia University, Morgantown, West Virginia, May 1974.

7. Michalski, R. S., and McCormick, B. H., "Interval Generalization of Switching Theory", Report number 442, Department of Computer Science, University of Illinois, Urbana, May 1971.

8. Michalski, R. S. and Larson, J. B., "Selection of Most Representative Training Examples and Incremental Generation of VL_1 Hypotheses: the underlying methodology and the description of programs ESEL and AQ11," Report Number 867, Department of Computer Science, University of Illinois, Urbana, May 1978.

Appendix A

CYBER Specific Operating Instructions

AQLISP is available in two forms on the CYBER - a LISP core-image overlay, and the original LISP source files. The core image is a pre-built AQLISP system that immediately invokes (AQVAL) as the top level function. It is recommended for users that do not require a change to CRITERIA&TOLERANCELIST, access to internal routines, or to LISP itself. Using the core-image significantly reduces overhead when loading AQLISP.

Currently, the AQLISP system requires a RFL of at least 140000B, but larger (160000B - 200000B) settings are recommended to reduce garbage collector time. The source is written for the UT LISP interpreter, which may be accessed by "GRAB,LISP".

The files for AQLISP are currently stored in user number 3RSMRSM. For accessing th core image version, a procedure file is available. Type:

```
/GET,AQBEGIN/UN=3RSMRSM
/-AQBEGIN
```

which loads LISP and obtains a local copy of the core image file "AQCORE". To start the interactive system, type:

```
    /-AQRUN
or /-AQRUN(CMD=<inputfile>)
```

If the "CMD=" parameter is supplied, AQLISP will read all input from the specified file, rather than the terminal, which allows files containing AQ commands to be prepared in advance. Otherwise, AQLISP begins execution and prompts the user as specified in section 2. Using the core image in this manner also produces a local file 'AQLOGGR', which contains a copy of all commands and prompts issued during the run.

To access the source files, you must load LISP, and type

```
/GET,AQLISP,LISPLIB/UN=3RSMRSM.
```

The incantation to LISP is

```
/RWF.
/LISP,C,E,K=160K,B=2000,A=85,AQLISP,LISPLIB.
```

At the top level of LISP, *EVAL:, use SETQ to initialize the parameters described in 3.3, and then issue (AQVAL) to *EVAL:.

The word size on the CYBER LISP is 60 bits, so at most 60 nominal values may be specified. However, the maximum integer value for Linear variables is 2^{48}, because of arithmetic restrictions imposed by CYBER LISP. Atomic names may be up to 30 display characters long, but if lower case is used each lower case character takes two display characters. Case is significant in CYBER LISP.

Appendix B-1

Source Listing

AQ Functions

Function: #COVERED

Purpose: Determines the number of events in EVENTS covered by COMP.

Calls:
INCLUDES

Function Listing:

```
(#COVERED
 (LAMBDA (COMP EVENTS **DUMMY**)
 (MINUS
  (PROG (**LST1 EV **VAL)
        (SETQ **LST1 EVENTS)
        (SETQ **VAL 0)
   **LP (SETQ EV (CAR (OR (LISTP **LST1) (GO **OUT))))
        (AND (INCLUDES COMP EV) (SETQ **VAL (ADD1 **VAL)))
 **ITERATE (SETQ **LST1 (CDR **LST1))
        (GO **LP)
  **OUT (RETURN **VAL)))))
```

Function: ABOVE

Calls:
ABOVE
PARENTS

Called By:
ABOVE

Function Listing:

```
(ABOVE
 (LAMBDA (VAL VALLIST VAR)
 (PROG (**LST1 **VAL V)
       (SETQ **LST1 VALLIST)
  **LP (SETQ V (CAR (OR (LISTP **LST1) (GO **OUT))))
       (COND ((OR (EQ VAL V) (ABOVE VAL (PARENTS V VAR) VAR))
              (SETQ **VAL V)
              (GO **OUT)))
 **ITERATE (SETQ **LST1 (CDR **LST1))
       (GO **LP)
 **OUT (RETURN **VAL))))
```

Function: AQ

Purpose: Top level function of AQ algorithm. Finds a cover of events in ELIST that do not cover any events in FLIST. If DISJOINTCOMPLEXFLAG is T, all complexes in the cover will be disjoint. Returns a list with first element being the number of iterations during part 2 of the algorithm, and the second element a list of complexes that compose the cover.

Calls:

STAR
KNOCKOUT
BESTCOMP

Called By:

INTERSECTINGCOVERS
DISJOINTCOMPLEXCOVER
DISJOINTCOVERS
INCREMENTALGEN
ORDEREDCOVER
PPAQ

Function Listing:

```
(AQ
 (LAMBDA (ELIST FLIST DISJOINTCOMPLEXFLG)
 (PROG (**VAL SMALLELIST UNCOVEREDELIST COVER DEL COUNTDEL)
       (SETQ SMALLELIST (COPY ELIST))
       (SETQ UNCOVEREDELIST (COPY ELIST))
       (SETQ DEL 0)
  **LP (COND ((NOT SMALLELIST) (GO **OUT)))
       (SETQ EVENT (CAR SMALLELIST))
       (SETQ STAR
             (STAR EVENT
                   (COND (DISJOINTCOMPLEXFLG (APPEND COVER FLIST))
                         (T FLIST))
                   ELIST
                   UNCOVEREDELIST))
       (MAPC STAR
             (FUNCTION
              (LAMBDA (COMP)
              (SETQ SMALLELIST (KNOCKOUT COMP SMALLELIST)))))
       (SETQ COVER (CONS (BESTCOMP STAR ELIST UNCOVEREDELIST) COVER))
       (SETQ UNCOVEREDELIST (KNOCKOUT (CAR COVER) UNCOVEREDELIST))
       (COND (COUNTDEL (SETQ DEL (PLUS DEL 1))))
**ITERATE (GO **LP)
**OUT (COND ((NULL UNCOVEREDELIST) (RETURN (LIST DEL COVER)))
            (T (SETQ COUNTDEL T)
               (SETQ SMALLELIST UNCOVEREDELIST)
               (GO **ITERATE)))
      (RETURN **VAL))))
```

Function: AQVAL

Purpose: Interactive driver for AQ function.

Calls:
READVARIABLES
READEVENTS
SHOWCOVERS
INTERSECTINGCOVERS

Function Listing:

```
(AQVAL
 (LAMBDA NIL
 (PROG (CLASSNAMES)
       (READVARIABLES)
       (SETQ CLASSNAMES (READEVENTS))
       (SETQ CLASSLIST (MAPCAR CLASSNAMES (FUNCTION EVENTS)))
    LP (SHOWCOVERS
        (SELECT
         (ASKUSER NIL
                  NIL
                  '((TERPRI) (TERPRI) MODE OF OPERATION IS: _ )
                  '(INTERSECTINGCOVERS NONINTERSECTINGCOVERS
                   DISJOINTCOMPLEXES SEQUENTIAL)
                  NIL)
         ('INTERSECTINGCOVERS (INTERSECTINGCOVERS CLASSLIST))
         ('NONINTERSECTINGCOVERS (DISJOINTCOVERS CLASSLIST))
         ('DISJOINTCOMPLEXES (DISJOINTCOMPLEXCOVER CLASSLIST))
         ('SEQUENTIAL (ORDEREDCOVER CLASSLIST))
         (SHOULDNT))
        CLASSNAMES)
       (SELECT
        (ASKUSER NIL
                 NIL
                 '(DO YOU WANT TO TRY ANOTHER MODE?)
                 '(YES NO)
                 T)
        ('YES (GO LP))
        ('NO (RETURN (RECLAIM)))
        (SHOULDNT)))))
```

Function: BESTCOMP

Purpose: Finds 'best' complex from a list of complexes (STAR) given the list of events to cover (POSITIVEEVENTS) and those events yet to be covered (UNCOVEREDEVENTS).

Calls:
TRUNCATE

Called By:
AQ
INCREMENTALGEN

Function Listing:

```
(BESTCOMP
 (LAMBDA (STAR POSITIVEEVENTS UNCOVEREDEVENTS)
 (CAR
  (TRUNCATE STAR 1 1 CRITERIA&TOLERANCELIST POSITIVEEVENTS
  UNCOVEREDEVENTS))))
```

Function: BESTN

Purpose: Returns the list of the first N elements of list LL.

Called By:
TRUNCATE

Function Listing:

```
(BESTN
 (LAMBDA (LL N)
 (PROG (**LST2 **VAL I **END L **TEM1 **TEM2)
       (SETQ **LST2 LL)
       (SETQ I 1)
       (SETQ **END N)
  **LP (SETQ L (CAR (OR (LISTP **LST2) (GO **OUT))))
       (COND ((GREATERP I **END) (GO **OUT)))
       (SETQ **TEM1 L)
       (COND (**TEM2 (RPLACD **TEM2 (SETQ **TEM2 (LIST **TEM1))))
             (T (SETQ **VAL (SETQ **TEM2 (LIST **TEM1)))))
**ITERATE (SETQ I (PLUS I 1))
       (SETQ **LST2 (CDR **LST2))
       (GO **LP)
**OUT (RETURN **VAL))))
```

Function: CANCELNOMINAL

Purpose: Determine if any values in NEWNOMINAL, a full-word, are in the nominal portion of nominal selector SEL.

Calls:
MASK

Function Listing:

```
(CANCELNOMINAL
 (LAMBDA (NEWNOMINAL SEL)
 (ZEROP (LOGAND (MASK SEL) NEWNOMINAL))))
```

Function: COVERINCLUDES

Purpose: Returns the first complex in COVER that includes EVENT's points, or NIL otherwise.

Calls:
INCLUDES

Called By:
INCREMENTALGEN

Function Listing:

```
(COVERINCLUDES
 (LAMBDA (COVER EVENT)
 (PROG (**LST1 **VAL COMP)
       (SETQ **LST1 COVER)
  **LP (SETQ COMP (CAR (OR (LISTP **LST1) (GO **OUT))))
       (COND ((INCLUDES COMP EVENT) (SETQ **VAL COMP) (GO **OUT)))
 **ITERATE (SETQ **LST1 (CDR **LST1))
       (GO **LP)
 **OUT (RETURN **VAL))))
```

Function: CUTSTAR

Purpose: Trim a star STAR to those complexes that are optimal according to function CRITFN applied with TOLERANCE.

Called By:
TRUNCATE

Function Listing:

```
(CUTSTAR
 (LAMBDA (STAR CRITFN TOLERANCE POSITIVEEVENTS UNCOVEREDEVENTS)
 (PROG (PLIST MAX MIN)
       (SETQ PLIST
             (MAPCAR STAR
                     (FUNCTION
                      (LAMBDA (COMP)
                      (PROGN
                       (SETQ VAL
                             (APPLY CRITFN
                                    (LIST COMP POSITIVEEVENTS
                                    UNCOVEREDEVENTS)))
                       (COND ((OR (NULL MAX) (GREATERP VAL MAX))
                              (SETQ MAX VAL)))
                       (COND ((OR (NULL MIN) (LESSP VAL MIN))
                              (SETQ MIN VAL)))
                       (CONS VAL COMP))))))
      (RETURN
       (PROG (**LST1 **VAL PAIR TOL **TEM1 **TEM2)
             (SETQ **LST1 PLIST)
             (SETQ TOL
                   (PLUS (TIMES TOLERANCE (DIFFERENCE MAX MIN)) MIN))
        **LP (SETQ PAIR (CAR (OR (LISTP **LST1) (GO **OUT))))
             (COND ((GREATERP (CAR PAIR) TOL) (GO **ITERATE)))
             (SETQ **TEM1 (CDR PAIR))
             (COND (**TEM2
                    (RPLACD **TEM2 (SETQ **TEM2 (LIST **TEM1))))
                   (T (SETQ **VAL (SETQ **TEM2 (LIST **TEM1)))))
   **ITERATE (SETQ **LST1 (CDR **LST1))
             (GO **LP)
       **OUT (RETURN **VAL))))))
```

Function: DISJOINTCOMPLEXCOVER

Purpose: Apply AQ to produce disjoint complexes in the cover.
(Interactive system).

Calls:
AQ

Function Listing:

```
(DISJOINTCOMPLEXCOVER
 (LAMBDA (CLASSES)
 (PROG (**LST1 **VAL CLASS **TEM1 **TEM2)
       (SETQ **LST1 CLASSES)
  **LP (SETQ CLASS (CAR (OR (LISTP **LST1) (GO **OUT))))
       (SETQ **TEM1
             (AQ CLASS
                 (APPEND
                  (MAPCON **VAL
                          (FUNCTION
                           (LAMBDA (C)
                           (PROGN (SETQ C (CAR C)) (APPEND (CADR C))))
                                                                  ))
                  (APPENDX (REMOVE CLASS CLASSES)))
                 T))
       (COND (**TEM2 (RPLACD **TEM2 (SETQ **TEM2 (LIST **TEM1))))
             (T (SETQ **VAL (SETQ **TEM2 (LIST **TEM1)))))
 **ITERATE (SETQ **LST1 (CDR **LST1))
       (GO **LP)
 **OUT (RETURN **VAL))))
```

Function: DISJOINTCOVERS

Purpose: Apply AQ to produce disjoint (but with possibly intersecting interval) covers. (Interactive system).

Calls:
AQ

Function Listing:

```
(DISJOINTCOVERS
 (LAMBDA (CLASSES)
 (PROG (**LST1 **VAL CLASS **TEM1 **TEM2)
       (SETQ **LST1 CLASSES)
  **LP (SETQ CLASS (CAR (OR (LISTP **LST1) (GO **OUT))))
       (SETQ **TEM1
             (AQ CLASS
                 (APPEND
                  (MAPCON **VAL
                          (FUNCTION
                           (LAMBDA (V)
                           (PROGN (SETQ V (CAR V)) (APPEND (CADR V))))
                                                                  ))
                  (APPENDX (REMOVE CLASS CLASSES)))
                 NIL))
       (COND (**TEM2 (RPLACD **TEM2 (SETQ **TEM2 (LIST **TEM1))))
             (T (SETQ **VAL (SETQ **TEM2 (LIST **TEM1)))))
**ITERATE (SETQ **LST1 (CDR **LST1))
       (GO **LP)
**OUT (RETURN **VAL))))
```

Function: EARLIER

Purpose: Used to order selectors within a complex. Sorts by ADDRESS of selector cell.

Called By:
INCLUDESSELECTORS
NEWCOMPLEXFROMLINEAR
EARLIERVAR

Function Listing:

```
(EARLIER (LAMBDA (X Y) (LESSP (ADDR X) (ADDR Y))))
```

Function: EPSILON

Called By:
EXTENDAGAINSTL
LINEARPRODUCTDIFREL

Function Listing:

```
(EPSILON (LAMBDA (VAR) (OR (GET VAR 'EPSILON) 1)))
```

Function: EQUALNOMINAL

Purpose: Determine if two NOMINAL fullwords are equal.

Function Listing:

```
(EQUALNOMINAL (LAMBDA (X Y) (EQN X Y)))
```

Function: EXTENDAGAINST

Purpose: Build a new complex by extending cover C1 against C2.

Calls:
EXTENDAGAINSTNOMINAL
EXTENDAGAINSTLINEAR
EXTENDAGAINSTSTRUCTURE

Called By:
STAR

Function Listing:

```
(EXTENDAGAINST
 (LAMBDA (C1 C2)
 (APPEND (EXTENDAGAINSTNOMINAL C1 C2)
         (APPEND (EXTENDAGAINSTLINEAR C1 C2)
                 (EXTENDAGAINSTSTRUCTURE C1 C2)))))
```

Function: EXTENDAGAINSTL

Purpose: Extends the range of the linear part of selector ES aganst linear part of selector FS. Produces a linear selector list.

Calls:

HIGH
LOW
BUILDLINEAR
VAR
EPSILON
MAXBOUND

Called By:

EXTENDAGAINSTLINEAR

Function Listing:

```
(EXTENDAGAINSTL
 (LAMBDA (ES FS)
 (COND ((LESSP (HIGH ES) (LOW FS))
        (BUILDLINEAR (VAR ES)
                     '=
                     0
                     (DIFFERENCE (LOW FS) (EPSILON (VAR ES)))))
       (T
        (BUILDLINEAR (VAR ES)
                     '=
                     (PLUS (HIGH FS) (EPSILON (VAR FS)))
                     (MAXBOUND (VAR FS)))))))
```

Function: EXTENDAGAINSTLINEAR

Purpose: Generates a star of linear-selector-only complexes that are extensions of elements in complex E against elements in complex f.

Calls:

LINEAR
FINDSELECTOR
VAR
LOW
BUILDCOMP
EXTENDAGAINSTL

Called By:

EXTENDAGAINST

Function Listing:

```
(EXTENDAGAINSTLINEAR
 (LAMBDA (E F)
 (PROG (**LST1 **VAL ELS FLS LF **TEM1 **TEM2)
       (SETQ **LST1 (LINEAR E))
       (SETQ LF (LINEAR F))
  **LP (SETQ ELS (CAR (OR (LISTP **LST1) (GO **OUT))))
       (COND ((NOT
                (AND (SETQ FLS (FINDSELECTOR LF (VAR ELS)))
                     (NOT (EQN (LOW ELS) (LOW FLS)))))
              (GO **ITERATE)))
       (SETQ **TEM1
             (BUILDCOMP NIL (LIST (EXTENDAGAINSTL ELS FLS)) NIL))
       (COND (**TEM2 (RPLACD **TEM2 (SETQ **TEM2 (LIST **TEM1))))
             (T (SETQ **VAL (SETQ **TEM2 (LIST **TEM1)))))
 **ITERATE (SETQ **LST1 (CDR **LST1))
       (GO **LP)
 **OUT (RETURN **VAL))))
```

Function: EXTENDAGAINSTNOMINAL

Purpose: Builds a star of nominal-selector-only complexes that are extensions of C1 against C2.

Calls:

NOMINAL
NEGATION
MASKLOGAND
BUILDCOMP
MASKLOGOR
MASKLOGXOR

Called By:

EXTENDAGAINST

Function Listing:

```
(EXTENDAGAINSTNOMINAL
 (LAMBDA (C1 C2)
 (PROG (**LST1 **VAL MASK N1 N2INVERSE **TEM1 **TEM2)
       (SETQ **LST1 MASKLIST)
       (SETQ N1 (NOMINAL C1))
       (SETQ N2INVERSE (NEGATION (NOMINAL C2)))
  **LP (SETQ MASK (CAR (OR (LISTP **LST1) (GO **OUT))))
       (COND ((ZEROP (MASKLOGAND MASK N1 N2INVERSE)) (GO **ITERATE)))
       (SETQ **TEM1
             (BUILDCOMP
              (MASKLOGOR (MASKLOGXOR MASK 7777777777777777777Q)
                         (MASKLOGAND MASK N2INVERSE NIL))
              NIL
              NIL))
       (COND (**TEM2 (RPLACD **TEM2 (SETQ **TEM2 (LIST **TEM1))))
             (T (SETQ **VAL (SETQ **TEM2 (LIST **TEM1)))))
**ITERATE (SETQ **LST1 (CDR **LST1))
       (GO **LP)
**OUT (RETURN **VAL))))
```

Function: EXTENDAGAINSTSTRUCTURE

Calls:
STRUCTURE
FINDSELECTOR
VAR
VALS
BUILDCOMP
BUILDSTRUCTURE
EXTENDAGAINSTSTRUCVAL

Called By:
EXTENDAGAINST

Function Listing:

```
(EXTENDAGAINSTSTRUCTURE
 (LAMBDA (E F)
 (PROG (**LST1 **VAL ES FS SF **TEM1 **TEM2)
       (SETQ **LST1 (STRUCTURE E))
       (SETQ SF (STRUCTURE F))
  **LP (SETQ ES (CAR (OR (LISTP **LST1) (GO **OUT))))
       (COND ((NOT
                (AND (SETQ FS (FINDSELECTOR SF (VAR ES)))
                     (NOT (EQUAL (VALS ES) (VALS FS)))))
              (GO **ITERATE)))
       (SETQ **TEM1
             (BUILDCOMP NIL
                        NIL
                        (LIST
                         (BUILDSTRUCTURE (VAR ES)
                                         '=
                                         (EXTENDAGAINSTSTRUCVAL ES FS))
                                                              )))
       (COND (**TEM2 (RPLACD **TEM2 (SETQ **TEM2 (LIST **TEM1))))
             (T (SETQ **VAL (SETQ **TEM2 (LIST **TEM1)))))
 **ITERATE (SETQ **LST1 (CDR **LST1))
       (GO **LP)
 **OUT (RETURN **VAL))))
```

Function: EXTENDAGAINSTSTRUCVAL

Calls:
VALS
SIBLINGS
VAR
PARENT
ALLPARENTS

Called By:
EXTENDAGAINSTSTRUCTURE

Function Listing:

```
(EXTENDAGAINSTSTRUCVAL
 (LAMBDA (ES FS)
 (COND ((MEMBER (VALS ES) (SIBLINGS (VALS FS) (VAR FS)))
        (SIBLINGS (VALS FS) (VAR FS)))
       (T
        (PROG (**VAL VAR EV EVP FVPARENTS)
              (SETQ VAR (VAR ES))
              (SETQ EV (VALS ES))
              (SETQ EVP (PARENT (VALS ES) (VAR ES)))
              (SETQ FVPARENTS (ALLPARENTS (VALS FS) (VAR FS)))
         **LP (COND ((MEMBER EVP FVPARENTS) (GO **OUT)))
              (SETQ EV EVP)
              (SETQ EVP (PARENT EVP VAR))
   **ITERATE (GO **LP)
       **OUT (RETURN EV))))))
```

Function: FINDSELECTOR

Purpose: Locates the selector in the list SELLIST that defines a value/interval for variable VAR.

Calls:
VAR

Called By:
EXTENDAGAINSTLINEAR
EXTENDAGAINSTSTRUCTURE
RESTRICT

Function Listing:

```
(FINDSELECTOR
 (LAMBDA (SELLIST VAR)
 (PROG (**LST1 **VAL SEL)
       (SETQ **LST1 SELLIST)
  **LP (SETQ SEL (CAR (OR (LISTP **LST1) (GO **OUT))))
       (COND ((EQ (VAR SEL) VAR) (RETURN SEL)))
**ITERATE (SETQ **LST1 (CDR **LST1))
       (GO **LP)
**OUT (RETURN **VAL))))
```

Function: INBOUNDS

Purpose: Determines if linear selector OUTERSEL contains lin. sel. INNERSEL.

Calls:
RELATION

Function Listing:

```
(INBOUNDS
 (LAMBDA (OUTERSEL INNERSEL)
 (SELECT (RELATION OUTERSEL)
         ('=
          (SELECT (RELATION INNERSEL)
                  ('=
                   (AND (NOT (LESSP (LOW INNERSEL) (LOW OUTERSEL)))
                        (NOT
                         (GREATERP (HIGH INNERSEL) (HIGH OUTERSEL)))))
                  (SHOULDNT)))
         ('#
          (OR (LESSP (HIGH INNERSEL) (LOW OUTERSEL))
              (GREATERP (LOW INNERSEL) (HIGH OUTERSEL))))
         (SHOUDNT))))
```

Function: INCLUDES

Purpose: Determines if complx COMP includes all points in EVENT.

Calls:
INCLUDESNOMINAL
NOMINAL
INCLUDESLINEAR
LINEAR
INCLUDESSTRUCTURE
STRUCTURE

Called By:
#COVERED
RESTRICT
KNOCKOUT
COVERINCLUDES
INCREMENTALGEN

Function Listing:

```
(INCLUDES
 (LAMBDA (COMP EVENT)
 (AND (INCLUDESNOMINAL (NOMINAL COMP) (NOMINAL EVENT))
      (INCLUDESLINEAR (LINEAR COMP) (LINEAR EVENT))
      (INCLUDESSTRUCTURE (STRUCTURE COMP) (STRUCTURE EVENT)))))
```

Function: INCLUDESLINEAR

Purpose: Returns T if linear selector in list L1 includes all points defined by selectors in L2.

Calls:
INCLUDESSELECTORS

Called By:
INCLUDES

Function Listing:

```
(INCLUDESLINEAR
 (LAMBDA (L1 L2)
 (INCLUDESSELECTORS L1 L2 (FUNCTION INBOUNDS))))
```

Function: INCLUDESNOMINAL

Purpose: Returns T if nominal selector in N1 contains all points in N2.

Calls:
MASKEQUAL
MASKLOGAND

Called By:
INCLUDES

Function Listing:

```
(INCLUDESNOMINAL
 (LAMBDA (N1 N2)
 (OR (EQN N1 7777777777777777777Q)
     (MASKEQUAL (MASKLOGAND N1 N2 NIL) N2))))
```

Function: INCLUDESSTRUCSEL

Calls:
RELATION

Function Listing:

```
(INCLUDESSTRUCSEL
 (LAMBDA (SOUTER SINNER)
 (SELECT (RELATION SOUTER)
         ('=
          (SELECT (RELATION SINNER)
                  ('=
                   (COND ((LISTP (VALS SOUTER))
                          (COND ((LISTP (VALS SINNER))
                                 (PROG (**LST1 INVAL **VAL)
                                       (SETQ **LST1 (VALS SINNER))
                                       (SETQ **VAL T)
                                  **LP (SETQ INVAL
                                             (CAR
                                             (OR (LISTP **LST1) (GO **OUT))
                                                                 ))
                                       (COND ((NULL
                                             (PROG (**LST1 **VAL OUTVAL)
                                             (SETQ **LST1 (VALS SOUTER))
                                        **LP (SETQ OUTVAL
                                             (CAR
                                             (OR (LISTP **LST1) (GO **OUT))
                                                                 ))
                                             (COND ((ABOVE OUTVAL INVAL)
                                             (SETQ **VAL OUTVAL)
                                             (GO **OUT)))
                                 **ITERATE (SETQ **LST1 (CDR **LST1))
                                             (GO **LP)
                                     **OUT (RETURN **VAL)))
                                             (SETQ **VAL NIL)
                                             (GO **OUT)))
                                **ITERATE (SETQ **LST1 (CDR **LST1))
                                          (GO **LP)
                                    **OUT (RETURN **VAL)))
                                   (T
                                    (PROG (**LST1 **VAL OUTVAL)
                                          (SETQ **LST1 (VALS SOUTER))
                                     **LP (SETQ OUTVAL
                                             (CAR
                                             (OR (LISTP **LST1) (GO **OUT))
                                                                 ))
```

```
                                        (COND ((ABOVE OUTVAL INVAL)
                                         (SETQ **VAL OUTVAL)
                                         (GO **OUT)))
                                    **ITERATE (SETQ **LST1 (CDR **LST1))
                                         (GO **LP)
                                       **OUT (RETURN **VAL)))))
                                ((LISTP (VALS SINNER))
                                 (PROG (**LST1 INVAL OUTVAL **VAL)
                                       (SETQ **LST1 (VALS SINNER))
                                       (SETQ OUTVAL (VALS SOUTER))
                                       (SETQ **VAL T)
                                  **LP (SETQ INVAL
                                             (CAR
                                              (OR (LISTP **LST1) (GO **OUT)))
                                                                  )
                                       (COND ((NULL (ABOVE OUTVAL INVAL))
                                              (SETQ **VAL NIL)
                                              (GO **OUT)))
                                 **ITERATE (SETQ **LST1 (CDR **LST1))
                                           (GO **LP)
                                      **OUT (RETURN **VAL)))
                                 ((ABOVE (VALS SOUTER) (VALS SINNER)))))
                            ('# (SHOULDNT))
                            (SHOULDNT)))
                    ('# (SHOULDNT))
                    (SHOULDNT))))
```

Function: INCLUDESSTRUCTURE

Calls:

INCLUDESSELECTORS

Called By:

INCLUDES

Function Listing:

```
(INCLUDESSTRUCTURE
 (LAMBDA (OUTERSTRUC INNERSTRUC)
 (INCLUDESSELECTORS OUTERSTRUC INNERSTRUC (FUNCTION INCLUDESSTRUCREL))
                                                                  ))
```

Function: INCREMENTALGEN

Calls:

SETEVENTS
EVENTS
COVERINCLUDES
FORMULA
INCLUDES
LISTDIFFERENCE
MULTIPLY
NEGATECOMPLEX
BESTCOMP
SETFORMULA
AQ

Function Listing:

```
(INCREMENTALGEN
 (LAMBDA (EVENT CLASSNAME CLASSNAMELIST)
 (SETEVENTS CLASSNAME (CONS EVENT (EVENTS CLASSNAME)))
 (PROG (**LST1 **VAL V2 INCREMENTALGENTHRESHOLD EVENTCOVER
       COVEREDV2EVENTS REDUCEDCOVER NEWCOVER V2COVER)
       (SETQ **LST1 (REMOVE CLASSNAME CLASSNAMELIST))
       (SETQ INCREMENTALGENTHRESHOLD
             (OR (NUMBERP (EVALV 'INCREMENTALGENTHRESHOLD)) 0))
  **LP (SETQ V2 (CAR (OR (LISTP **LST1) (GO **OUT))))
       (COND ((NOT (COVERINCLUDES (FORMULA V2) EVENT)) (GO **ITERATE))
                                                       )
       (SETQ EVENTCOVER
             (SUBSET (FORMULA V2)
                     (FUNCTION
                      (LAMBDA (COVERCOMP)
                      (INCLUDES COVERCOMP EVENT)))
                     NIL))
       (SETQ V2COVER (LISTDIFFERENCE (FORMULA V2) EVENTCOVER))
       (SETQ COVEREDV2EVENTS
             (SUBSET (EVENTS V2)
                     (FUNCTION
                      (LAMBDA (EV)
                      (AND
                       (PROG (**LST1 **VAL COMP)
                             (SETQ **LST1 EVENTCOVER)
                        **LP (SETQ COMP
                                   (CAR (OR (LISTP **LST1) (GO **OUT))
                                                       ))
                             (COND ((INCLUDES COMP EV)
                                    (SETQ **VAL COMP)
```

```
                                    (SETQ **VAL
                                          (SETQ **TEM2 (LIST **TEM1)))))
                  **ITERATE (SETQ **LST1 (CDR **LST1))
                            (GO **LP)
                        **OUT (RETURN **VAL)))
                 (SETQ NEWCOVER
                       (CONS
                        (BESTCOMP REDUCEDCOVER COVEREDV2EVENTS
                        COVEREDV2EVENTS)
                        NEWCOVER))     .
                 (COND ((SETQ COVEREDV2EVENTS
                              (SUBSET COVEREDV2EVENTS
                                      (FUNCTION
                                       (LAMBDA (EV)
                                       (NOT (INCLUDES (CAR NEWCOVER) EV))
                                                      ))))
                        (GO LP))
                       (T (SETFORMULA V2 (APPEND NEWCOVER V2COVER)))))
                (T
                 (SETQ NEWCOVER
                       (APPEND
                        (AQ COVEREDV2EVENTS
                            (MAPCON CLASSNAMELIST
                                    (FUNCTION
                                     (LAMBDA (CN)
                                     (PROGN (SETQ CN (CAR CN))
                                            (AND (NOT (EQ CN V2))
                                            (APPEND
                                            (OR (EVENTS CN) (FORMULA CN)))
                                                          ))))))))))
**ITERATE (SETQ **LST1 (CDR **LST1))
      (GO **LP)
**OUT (RETURN **VAL))
(COND ((COVERINCLUDES (FORMULA CLASSNAME) EVENT))
      (T
       (SETFORMULA CLASSNAME
                   (APPEND
                    (CADAR
                     (AQ (LIST EVENT)
                         (MAPCON CLASSNAMELIST
                                 (FUNCTION
                                  (LAMBDA (CN)
                                  (PROGN (SETQ CN (CAR CN))
                                         (AND (NOT (EQ CN CLASSNAME))
                                         (APPEND
                                         (OR (EVENTS CN) (FORMULA CN)))
                                                       ))))))))
                   (FORMULA CLASSNAME)))))))
```

```
                                (GO **OUT)))
                    **ITERATE (SETQ **LST1 (CDR **LST1))
                              (GO **LP)
                        **OUT (RETURN **VAL))
                        (PROG (**LST1 COMP **VAL)
                              (SETQ **LST1 V2COVER)
                              (SETQ **VAL T)
                         **LP (SETQ COMP
                                    (CAR (OR (LISTP **LST1) (GO **OUT))
                                                                    ))
                                   (COND ((NULL (NOT (INCLUDES COMP EV)))
                                          (SETQ **VAL NIL)
                                          (GO **OUT)))
                    **ITERATE (SETQ **LST1 (CDR **LST1))
                              (GO **LP)
                        **OUT (RETURN **VAL)))))))
   (SETQ REDUCEDCOVER (MULTIPLY EVENTCOVER (NEGATECOMPLEX EVENT)))
LP (COND ((SETQ REDUCEDCOVER
                (PROG (**LST1 **VAL COMP **TEM1 **TEM2)
                      (SETQ **LST1 REDUCEDCOVER)
                 **LP (SETQ COMP
                            (CAR (OR (LISTP **LST1) (GO **OUT))))
                      (COND ((NOT
                               (PROG (**LST1 EV **VAL)
                                     (SETQ **LST1 COVEREDV2EVENTS)
                                     (SETQ **VAL 0)
                                **LP (SETQ EV
                                     (CAR
                                     (OR (LISTP **LST1) (GO **OUT))
                                                                  ))
                                     (COND ((GREATERP **VAL
                                     INCREMENTALGENTHRESHOLD)
                                     (GO **OUT))
                                     ((NOT (INCLUDES COMP EV))
                                     (GO **ITERATE)))
                                     (SETQ **VAL (PLUS **VAL 1))
                           **ITERATE (SETQ **LST1 (CDR **LST1))
                                     (GO **LP)
                               **OUT (RETURN
                                     (GREATERP **VAL
                                     INCREMENTALGENTHRESHOLD))))
                             (GO **ITERATE)))
                      (SETQ **TEM1 COMP)
                      (COND (**TEM2
                             (RPLACD **TEM2
                                     (SETQ **TEM2 (LIST **TEM1))))
                            (T
```

Function: INCLUDESSELECTORS

Purpose: Using function INCLUDESFN, decide if all selectors in list INSIDESELECTORS are contained within the range defined by OUTSIDESELECTORS.

Calls:
NON-TRIVIAL
VAR
EARLIER

Called By:
INCLUDESLINEAR
INCLUDESSTRUCTURE

Function Listing:

```
(INCLUDESSELECTORS
 (LAMBDA (OUTSIDESELECTORS INSIDESELECTORS INCLUDESFN)
 (PROG (**LST1 SEL **VAL)
       (SETQ **LST1 OUTSIDESELECTORS)
       (SETQ **VAL T)
  **LP (SETQ SEL (CAR (OR (LISTP **LST1) (GO **OUT))))
       (COND ((NOT (NON-TRIVIAL SEL)) (GO **ITERATE)))
       (COND ((NULL
                (PROG (**VAL)
                 **LP (COND ((OR (NULL INSIDESELECTORS)
                                 (EQ (VAR (CAR INSIDESELECTORS))
                                     (VAR SEL))
                                 (EARLIER (VAR SEL)
                                          (VAR (CAR INSIDESELECTORS))))
                            (GO **OUT)))
                      (SETQ INSIDESELECTORS (CDR INSIDESELECTORS))
           **ITERATE (GO **LP)
               **OUT (RETURN
                       (AND (EQ (VAR (CAR INSIDESELECTORS)) (VAR SEL))
                            (APPLY INCLUDESFN
                                   (LIST SEL (CAR INSIDESELECTORS))))))
                                                                    )
              (SETQ **VAL NIL)
              (GO **OUT)))
 **ITERATE (SETQ **LST1 (CDR **LST1))
       (GO **LP)
 **OUT (RETURN **VAL))))
```

Function: INTERSECTINGCOVERS

Purpose: Generates a cover of classes in an intersecting-cover mode. (Interactive system).

Calls:
AQ

Called By:
AQVAL

Function Listing:

```
(INTERSECTINGCOVERS
 (LAMBDA (CLASSES)
 (MAPCAR CLASSES
         (FUNCTION
          (LAMBDA (CLASS)
          (AQ CLASS (APPENDX (REMOVE CLASS CLASSES)) NIL))))))
```

Function: LINEARPRODUCT

Calls:
RELATION
HIGH
LOW
BUILDLINEAR
VAR
LINEARPRODUCTDIFREL

Function Listing:

```
(LINEARPRODUCT
 (LAMBDA (SEL1 SEL2)
 (PROG (REL LOW HIGH)
       (COND ((EQ (RELATION SEL1) '=)
              (COND ((EQ (RELATION SEL2) '=)
                     (COND ((OR (LESSP (HIGH SEL1) (LOW SEL2))
                                (GREATERP (LOW SEL1) (HIGH SEL2)))
                            (RETURN 'CANCEL)))
                     (COND ((LESSP (LOW SEL1) (LOW SEL2))
                            (SETQ LOW (LOW SEL2)))
                           (T (SETQ LOW (LOW SEL1))))
                     (COND ((GREATERP (HIGH SEL1) (HIGH SEL2))
                            (SETQ HIGH (HIGH SEL2)))
                           ((EQN LOW (LOW SEL1)) (RETURN 'ABSORB))
                           (T (SETQ HIGH (HIGH SEL1))))
                     (RETURN
                      (CONS 'NON-DISJUNCTION
                            (BUILDLINEAR (VAR SEL2) '= LOW HIGH))))
                    ((OR (NOT (LESSP (LOW SEL2) (HIGH SEL1)))
                         (NOT (GREATERP (HIGH SEL2) (LOW SEL1))))
                     (RETURN 'ABSORB))
                    (T (RETURN (LINEARPRODUCTDIFREL SEL2 SEL1)))))
             ((EQ (RELATION SEL2) '=)
              (RETURN (LINEARPRODUCTDIFREL SEL1 SEL2)))
             (T
              (SETQ LOW
                    (LOW
                     (COND ((LESSP (LOW SEL2) (LOW SEL1)) SEL2)
                           (T SEL1))))
              (COND ((GREATERP (HIGH SEL2) (HIGH SEL1))
                     (SETQ HIGH (HIGH SEL2)))
                    ((EQN LOW (LOW SEL1)) (RETURN 'ABSORB))
                    (T (SETQ HIGH (HIGH SEL1))))
              (RETURN
               (CONS 'NON-DISJUNCTION
                     (BUILDLINEAR (VAR SEL2) '# LOW HIGH)))))))))
```

Function: KNOCKOUT

Purpose: Removes all events in EVLIST that the cover COMP covers.

Calls:
INCLUDES

Called By:
AQ

Function Listing:

```
(KNOCKOUT
 (LAMBDA (COMP EVLIST)
 (SUBSET EVLIST (FUNCTION (LAMBDA (EV) (NOT (INCLUDES COMP EV)))) NIL)
                                                                  ))
```

Function: LISTDIFFERENCE

Called By:
INCREMENTALGEN

Function Listing:

```
(LISTDIFFERENCE
 (LAMBDA (XL YL)
 (SUBSET XL (FUNCTION (LAMBDA (X) (NOT (MEMB X YL)))))))
```

Function: MASK

Purpose: Returns the mask corresponding to the variable that is used in nominal selector SEL.

Called By:
CANCELNOMINAL
NEGATECOMPLEX
NOMINALPRODUCT

Function Listing:

```
(MASK
 (LAMBDA (SEL)
 (PROGN (SETQ SEL (LOGXOR 7777777777777777777Q SEL))
        (PROG (**LST1 **VAL MASK)
              (SETQ **LST1 MASKLIST)
         **LP (SETQ MASK (CAR (OR (LISTP **LST1) (GO **OUT))))
              (COND ((NOT (ZEROP (LOGAND MASK SEL))) (RETURN MASK)))
    **ITERATE (SETQ **LST1 (CDR **LST1))
              (GO **LP)
        **OUT (RETURN 77777777777777777777Q)))))
```

Function: LINEARPRODUCTDIFREL

Calls:
LOW
HIGH
BUILDLINEAR
VAR
EPSILON

Called By:
LINEARPRODUCT

Function Listing:

```
(LINEARPRODUCTDIFREL
 (LAMBDA (INEQSEL EQSEL)
 (PROG (LOW HIGH)
       (COND ((AND (LESSP (LOW EQSEL) (LOW INEQSEL))
                   (GREATERP (HIGH EQSEL) (HIGH INEQSEL)))
              (RETURN
               (LIST 'DISJUNCTION
                     (BUILDLINEAR (VAR EQSEL)
                                  '=
                                  (LOW EQSEL)
                                  (DIFFERENCE (LOW INEQSEL)
                                              (EPSILON (VAR EQSEL))))
                     (BUILDLINEAR (VAR EQSEL)
                                  '=
                                  (PLUS (HIGH INEQSEL)
                                        (EPSILON (VAR EQSEL)))
                                  (HIGH EQSEL))))))
       (SETQ LOW
             (COND ((LESSP (LOW EQSEL) (LOW INEQSEL)) (LOW EQSEL))
                   ((NOT (LESSP (HIGH INEQSEL) (HIGH EQSEL)))
                    (RETURN 'CANCEL))
                   ((LESSP (HIGH INEQSEL) (LOW EQSEL)) (RETURN EQSEL))
                   (T (PLUS (HIGH INEQSEL) (EPSILON (VAR EQSEL))))))
       (SETQ HIGH
             (COND ((GREATERP (HIGH EQSEL) (HIGH INEQSEL))
                    (HIGH EQSEL))
                   ((GREATERP (LOW INEQSEL) (HIGH EQSEL))
                    (RETURN EQSEL))
                   (T (DIFFERENCE (LOW INEQSEL) (EPSILON (VAR EQSEL)))
                                                                 )))
       (RETURN (BUILDLINEAR (VAR EQSEL) '= LOW HIGH)))))
```

Function: MULTIPLY

Calls:
PRODUCTSC
SELECTOROFTRIVIALCOMPLEX

Called By:
STAR
INCREMENTALGEN

Function Listing:

```
(MULTIPLY
 (LAMBDA (COMPSET SELSET)
 (PROG (**LST1 **VAL COMP)
       (SETQ **LST1 COMPSET)
  **LP (SETQ COMP (CAR (OR (LISTP **LST1) (GO **OUT))))
       (SETQ **VAL
             (NCONC **VAL
                    (PROG (**LST1 **VAL SEL PRODUCT)
                          (SETQ **LST1 SELSET)
                           **LP (SETQ SEL
                                      (CAR (OR (LISTP **LST1) (GO **OUT))))
                                (COND ((EQ
                                         (SETQ PRODUCT
                                               (PRODUCTSC
                                               (SELECTOROFTRIVIALCOMPLEX SEL)
                                               COMP
                                               NIL))
                                         'ABSORB)
                                        (GO **OUT)))
                          (SETQ **VAL (NCONC **VAL PRODUCT))
                **ITERATE (SETQ **LST1 (CDR **LST1))
                          (GO **LP)
                    **OUT (RETURN
                           (SELECT PRODUCT
                                   ('ABSORB (LIST COMP))
                                   **VAL)))))
 **ITERATE (SETQ **LST1 (CDR **LST1))
       (GO **LP)
 **OUT (RETURN **VAL))))
```

Function: MINLIST

Calls:
DOMINATE

Called By:
STRUCPRODUCT

Function Listing:

```
(MINLIST
 (LAMBDA (VALS1 VALS2 VAR)
 (PROGN
  (SETQ VALS1
        (MAPCON (OR (LISTP VALS1) (LIST VALS1))
                (FUNCTION
                 (LAMBDA (VAL1)
                 (PROGN (SETQ VAL1 (CAR VAL1))
                        (MAPCON (OR (LISTP VALS2) (LIST VALS2))
                                (FUNCTION
                                 (LAMBDA (VAL2)
                                 (PROGN (SETQ VALS2 (CAR VALS2))
                                        (COND ((DOMINATE VAL2 VAL1 VAR
                                        )
                                        (LIST VAL1))
                                        ((DOMINATE VAL1 VAL2 VAR)
                                        (LIST VAL2)))))))))))
 (SETQ VALS1 (INTERSECTION VALS1 VALS1))
 (COND ((CDR VALS1) VALS1)
       (T (CAR VALS1))))))
```

Function: NEGATELINEARSELECTOR

Calls:
BUILDLINEAR
VAR
RELATION
VALS

Called By:
NEGATECOMPLEX

Function Listing:

```
(NEGATELINEARSELECTOR
 (LAMBDA (SEL)
 (BUILDLINEAR (VAR SEL)
              (SELECT (RELATION SEL) ('= '#) ('# '=) (SHOULDNT))
              (VALS SEL))))
```

Function: NEGATESTRUCTURESELECTOR

Calls:
BUILDSTRUCTURE
VAR
RELATION
VALS

Called By:
NEGATECOMPLEX

Function Listing:

```
(NEGATESTRUCTURESELECTOR
 (LAMBDA (SEL)
 (BUILDSTRUCTURE (VAR SEL)
                 (SELECT (RELATION SEL) ('= '#) ('# '=) (SHOULDNT))
                 (VALS SEL))))
```

Function: NEGATECOMPLEX

Purpose: Returns the 'negative' of a complex, i.e., all points not within the complexes space.

Calls:

MASK
BUILDCOMP
MASKLOGOR
NEGATION
MASKLOGXOR
NOMINAL
LINEAR
NEGATELINEARSELECTOR
STRUCTURE
NEGATESTRUCTURESELECTOR

Called By:

INCREMENTALGEN

Function Listing:

```
(NEGATECOMPLEX
 (LAMBDA (COMPLEX)
 (APPEND
  (MAPCAR MASKLIST
          (FUNCTION
           (LAMBDA (MASK)
           (BUILDCOMP
            (MASKLOGOR (NEGATION MASK)
                       (MASKLOGXOR MASK (NOMINAL COMPLEX)))
            NIL
            NIL))))
 (APPEND
  (MAPCAR (LINEAR COMPLEX)
          (FUNCTION
           (LAMBDA (SEL)
           (BUILDCOMP NIL (NEGATELINEARSELECTOR SEL) NIL))))
  (MAPCAR (STRUCTURE COMPLEX)
          (FUNCTION
           (LAMBDA (SEL)
           (BUILDCOMP NIL NIL (NEGATESTRUCTURESELECTOR SEL)))))))))
```

Function: NUMBEROFSELECTORS

Purpose: Count the number of selectors in a complex COMP.

Calls:
NEGATION
NOMINAL
LINEAR
STRUCTURE

Function Listing:

```
(NUMBEROFSELECTORS
 (LAMBDA (COMP **DUMMY1** **DUMMY2**)
 (PLUS
  (PROG (**LST1 MASK NEGNOM **VAL)
        (SETQ **LST1 MASKLIST)
        (SETQ NEGNOM (NEGATION (NOMINAL COMP)))
        (SETQ **VAL 0)
   **LP (SETQ MASK (CAR (OR (LISTP **LST1) (GO **OUT))))
        (SETQ **VAL
              (PLUS **VAL
                    (COND ((ZEROP (LOGAND NEGNOM MASK)) 0)
                          (T 1))))
 **ITERATE (SETQ **LST1 (CDR **LST1))
        (GO **LP)
  **OUT (RETURN **VAL))
  (LENGTH (LINEAR COMP))
  (LENGTH (STRUCTURE COMP)))))
```

Function: ORDEREDCOVER

Calls:
AQ

Function Listing:

```
(ORDEREDCOVER
 (LAMBDA (CLASSES)
 (MAPLIST CLASSES
          (FUNCTION
           (LAMBDA (CLASSTAIL)
           (AQ (CAR CLASSTAIL) (APPENDX (CDR CLASSTAIL)) NIL))))))
```

Function: NEWCOMPLEXFROMLINEAR

Purpose: Builds a new complex with linear selector NEWSELECTOR substituted for OLDSELECTOR.

Calls:

BUILDCOMP
NOMINAL
LINEAR
EARLIER
VAR
STRUCTURE

Function Listing:

```
(NEWCOMPLEXFROMLINEAR
 (LAMBDA (COMP OLDSELECTOR NEWSELECTOR)
 (BUILDCOMP (NOMINAL COMP)
            (SORT
             (CONS NEWSELECTOR (REMOVE OLDSELECTOR (LINEAR COMP)))
             (FUNCTION (LAMBDA (X Y) (EARLIER (VAR X) (VAR Y)))))
            (STRUCTURE COMP))))
```

Function: NOMINALPRODUCT

Purpose: Finds the product (intersection) of all nominal points in SEL and NOM.

Calls:

MASK

Function Listing:

```
(NOMINALPRODUCT
 (LAMBDA (SEL NOM)
 (LOGOR (LOGAND SEL NOM)
        (LOGAND (LOGXOR 7777777777777777777Q (MASK SEL)) NOM))))
```

```
                                    (FUNCTION EARLIERVAR))
                                (STRUCTURE COMP))))))
        ('STRUCTURED
         (COND ((SETQ OLDSELECTOR
                      (FINDSELECTOR (STRUCTURE COMP) (VAR SEL)))
                (SELECT (SETQ PRODUCT (STRUCPRODUCT OLDSELECTOR SEL))
                        ('ABSORB 'ABSORB)
                        ('CANCEL NIL)
                        (LIST
                         (NEWCOMPLEXFROMSTRUCTURE COMP OLDSELECTOR
                         PRODUCT))))
               (T
                (LIST
                 (BUILDCOMP (NOMINAL COMP)
                            (LINEAR COMP)
                            (SORT
                             (APPEND (STRUCTURE COMP) (LIST SEL))
                             (FUNCTION EARLIERVAR)))))))
        (SHOULDNT))))
```

Function: PRODUCTSC1

Calls:

PRODUCTSC

Function Listing:

```
(PRODUCTSC1
 (LAMBDA (SEL COMP)
 (PROGN (SETQ SEL (PRODUCTSC SEL COMP))
        (SELECT SEL ('ABSORB COMP) SEL))))
```

Function: PRODUCTSC

Calls:

SELECTORTYPE

Called By:

MULTIPLY
PRODUCTSC1

Function Listing:

```
(PRODUCTSC
 (LAMBDA (SEL COMP PRODUCT)
 (SELECT (SELECTORTYPE SEL)
         ('NOMINAL
          (SETQ NEWNOMINAL (NOMINALPRODUCT SEL (NOMINAL COMP)))
          (COND ((EQUALNOMINAL NEWNOMINAL (NOMINAL COMP)) 'ABSORB)
                ((CANCELNOMINAL NEWNOMINAL SEL) NIL)
                (T
                 (LIST
                  (BUILDCOMP NEWNOMINAL
                             (LINEAR COMP)
                             (STRUCTURE COMP))))))
        ('LINEAR
         (COND ((SETQ OLDSELECTOR
                      (FINDSELECTOR (LINEAR COMP) (VAR SEL)))
                (SELECT
                 (SETQ PRODUCT (LINEARPRODUCT OLDSELECTOR SEL))
                 ('ABSORB 'ABSORB)
                 ('CANCEL NIL)
                 (COND ((EQ (CAR PRODUCT) 'DISJUNCTION)
                        (LIST
                         (NEWCOMPLEXFROMLINEAR COMP
                                               OLDSELECTOR
                                               (CADR PRODUCT))
                         (NEWCOMPLEXFROMLINEAR COMP
                                               OLDSELECTOR
                                               (CADDR PRODUCT))))
                       (T
                        (LIST
                         (NEWCOMPLEXFROMLINEAR COMP
                                               OLDSELECTOR
                                               (CDR PRODUCT)))))))
               (T
                (LIST
                 (BUILDCOMP (NOMINAL COMP)
                            (SORT (APPEND (LINEAR COMP) (LIST SEL))
```

Function: SELECTORTYPE

Purpose: Return the type of the selector SEL.

Calls:
NOMINALP
VAR

Called By:
PRODUCTSC

Function Listing:

```
(SELECTORTYPE
 (LAMBDA (SEL)
 (COND ((NOMINALP SEL) 'NOMINAL)
       ((AND (LISTP SEL) (GET (VAR SEL) 'DOMAINTYPE)))
       (T (SHOULDNT)))))
```

Function: STAR

Purpose: Generates the star of complxes of E against FLIST.

Calls:
BUILDCOMP
EXTENDAGAINST
TRUNCATE
MULTIPLY

Called By:
AQ
PPSTAR

Function Listing:

```
(STAR
 (LAMBDA (E FLIST POSITIVEEVENTS UNCOVEREDELIST)
 (PROG (**LST1 **VAL F PRODUCT)
       (SETQ **LST1 FLIST)
       (SETQ PRODUCT (LIST (BUILDCOMP NIL NIL NIL)))
  **LP (SETQ F (CAR (OR (LISTP **LST1) (GO **OUT))))
       (SETQ NEWPRODUCT (EXTENDAGAINST E F))
       (SETQ PRODUCT
             (TRUNCATE (MULTIPLY PRODUCT NEWPRODUCT)
                       MAXSTAR
                       CUTSTAR
                       CRITERIA&TOLERANCELIST
                       POSITIVEEVENTS
                       UNCOVEREDELIST))
  **ITERATE (SETQ **LST1 (CDR **LST1))
       (GO **LP)
  **OUT (RETURN PRODUCT))))
```

Function: RESTRICT

Calls:

INCLUDES
BUILDCOMP
MASKLOGOR
NOMINAL
MASKLOGAND
LINEAR
FINDSELECTOR
VAR
STRUCTURE

Called By:

TRUNCATE

Function Listing:

```
(RESTRICT
 (LAMBDA (COMPLEX RESTRICTION)
 (COND ((INCLUDES (CONDITION RESTRICTION) COMPLEX)
        (BUILDCOMP
         (MASKLOGOR (NOMINAL (RHS RESTRICTION))
                    (MASKLOGAND
                     (TRIVIALVARMASK (NOMINAL (RHS RESTRICTION)))
                     (NOMINAL COMPLEX)
                     NIL))
         (PROG (**LST1 **VAL SEL LINEAR)
               (SETQ **LST1 (LINEAR (RHS RESTRICTION)))
               (SETQ LINEAR (COPY (LINEAR COMPLEX)))
          **LP (SETQ SEL (CAR (OR (LISTP **LST1) (GO **OUT))))
               (SETQ LINEAR
                     (REMOVE (FINDSELECTOR LINEAR (VAR SEL)) LINEAR))
   **ITERATE (SETQ **LST1 (CDR **LST1))
               (GO **LP)
       **OUT (RETURN
                (SORT (APPEND (LINEAR (RHS RESTRICTION)) LINEAR)
                      (FUNCTION EARLIERVAR))))
         (PROG (**LST1 **VAL SEL STRUCTURE)
               (SETQ **LST1 (STRUCTURE (RHS RESTRICTION)))
               (SETQ STRUCTURE (COPY (STRUCTURE COMPLEX)))
          **LP (SETQ SEL (CAR (OR (LISTP **LST1) (GO **OUT))))
               (SETQ STRUCTURE
                     (REMOVE (FINDSELECTOR STRUCTURE (VAR SEL))
                             STRUCTURE))
   **ITERATE (SETQ **LST1 (CDR **LST1))
               (GO **LP)
       **OUT (RETURN
                (SORT (APPEND (STRUCTURE (RHS RESTRICTION)) STRUCTURE)
                      (FUNCTION EARLIERVAR))))))
     (T COMPLEX))))
```

Function: TRUNCATE

Purpose: Cuts (trims) the size of a star to CUTSTAR if it exceeds MAXSTAR in size, using the optimality criterion specified in C&TLIST. (see 3.3)

Calls:
RESTRICT
CUTSTAR
BESTN

Called By:
STAR
BESTCOMP

Function Listing:

```
(TRUNCATE
 (LAMBDA (STAR MAXSTAR CUTSTAR C&TLIST POSITIVEEVENTS UNCOVEREDEVENTS)
 (COND ((LESSP (LENGTH STAR) MAXSTAR) STAR)
       (T
        (PROG (**VAL STARTAIL)
              (SETQ STARTAIL STAR)
         **LP (COND ((NOT (LISTP STARTAIL)) (GO **OUT)))
              (RPLACA STARTAIL
                      (PROG (**LST1 **VAL RESTRICTION COMP)
                            (SETQ **LST1
                                  (LISTP (EVALV 'RESTRICTIONLIST)))
                            (SETQ COMP (CAR STARTAIL))
                             **LP (SETQ RESTRICTION
                                        (CAR (OR (LISTP **LST1) (GO **OUT)))
                                                                 )
                                  (SETQ COMP (RESTRICT COMP RESTRICTION))
                        **ITERATE (SETQ **LST1 (CDR **LST1))
                                  (GO **LP)
                             **OUT (RETURN COMP)))
   **ITERATE (SETQ STARTAIL (CDR STARTAIL))
              (GO **LP)
        **OUT (RETURN **VAL))
        (PROG (**LST1 **VAL C&T)
              (SETQ **LST1 C&TLIST)
         **LP (SETQ C&T (CAR (OR (LISTP **LST1) (GO **OUT))))
              (COND ((LESSP (LENGTH STAR) CUTSTAR) (GO **OUT)))
              (SETQ STAR
                    (CUTSTAR STAR
                             (CAR C&T)
                             (CADR C&T)
```

Function: STRUCPRODUCT

Calls:
MINLIST
VALS
VAR
BUILDSTRUCTURE

Function Listing:

```
(STRUCPRODUCT
 (LAMBDA (SEL1 SEL2)
 (COND ((SETQ MINLIST (MINLIST (VALS SEL1) (VALS SEL2) (VAR SEL1)))
        (COND ((NOT (EQUAL MINLIST (VALS SEL1)))
               (BUILDSTRUCTURE (VAR SEL1) '= MINLIST))
              (T 'ABSORB)))
       (T 'CANCEL))))
```

Appendix B-2

Data Structure Functions

Function: ADDLINEAR

Calls:

SETLINEAR
BUILDLINEAR
LINEAR

Function Listing:

```
(ADDLINEAR
 (LAMBDA (SELECTOR COMPLEX)
 (SETLINEAR COMPLEX
            (SORT
             (CONS
              (BUILDLINEAR (CAR SELECTOR)
                           (CADR SELECTOR)
                           (CADDR SELECTOR)
                           (CADDDR SELECTOR))
              (LINEAR COMPLEX))
             (FUNCTION EARLIERVAR)))))
```

```
                                POSITIVEEVENTS
                                UNCOVEREDEVENTS))
    **ITERATE (SETQ **LST1 (CDR **LST1))
              (GO **LP)
        **OUT (RETURN **VAL))
        (COND ((LESSP (LENGTH STAR) CUTSTAR) STAR)
              (T (BESTN STAR CUTSTAR)))))))
```

Function: ALLPARENTS

Calls:
PARENT

Called By:
EXTENDAGAINSTSTRUCVAL

Function Listing:

```
(ALLPARENTS
 (LAMBDA (NODE VAR)
 (PROG (**VAL **TEM1 **TEM2)
  **LP (COND ((NOT (SETQ NODE (PARENT NODE VAR))) (GO **OUT)))
       (SETQ **TEM1 NODE)
       (COND (**TEM2 (RPLACD **TEM2 (SETQ **TEM2 (LIST **TEM1))))
             (T (SETQ **VAL (SETQ **TEM2 (LIST **TEM1)))))
 **ITERATE (GO **LP)
 **OUT (RETURN **VAL))))
```

Function: BUILDCOMP

Calls:
TOTALMASK

Called By:
STAR
EXTENDAGAINSTNOMINAL
EXTENDAGAINSTLINEAR
EXTENDAGAINSTSTRUCTURE
RESTRICT
READVL1
NEGATECOMPLEX
NEWCOMPLEXFROMLINEAR
NEWCOMPLEXFROMSTRUCTURE

Function Listing:

```
(BUILDCOMP
 (LAMBDA (NOMINAL LINEAR STRUC)
 (LIST
  (LOGAND (TOTALMASK MASKLIST) (OR NOMINAL 7777777777777777777Q))
  LINEAR
  STRUC)))
```

Function: ADDNOMINAL

Calls:

SETNOMINAL
MASKLOGOR
MASKLOGAND
NOMINAL
NEGATION
GETMASK
BUILDMASK

Function Listing:

```
(ADDNOMINAL
 (LAMBDA (SELECTOR COMPLEX)
 (SETNOMINAL COMPLEX
             (MASKLOGOR
              (MASKLOGAND (NOMINAL COMPLEX)
                          (NEGATION (GETMASK (CAR SELECTOR)))
                          NIL)
              (COND ((EQ (CADR SELECTOR) '=)
                     (BUILDMASK (CAR SELECTOR) (CDDR SELECTOR)))
                    (T
                     (NEGATION
                      (BUILDMASK (CAR SELECTOR) (CDDR SELECTOR))))))))
                                                                  )
```

Function: ADDSTRUCTURED

Calls:

SETSTRUCTURED
BUILDSTRUCTURE
STRUCTURE

Function Listing:

```
(ADDSTRUCTURED
 (LAMBDA (SLECTOR COMPLEX)
 (SETSTRUCTURED COMPLEX
                (SORT
                 (CONS
                  (BUILDSTRUCTURE (CAR SELECTOR)
                                  (CADR SELECTOR)
                                  (CDDR SELECTOR))
                  (STRUCTURE COMPLEX))
                 (FUNCTION EARLIERVAR)))))
```

Function: BUILDMASK

Calls:
MASKLOGOR
MASKBIT

Called By:
ADDNOMINAL

Function Listing:

```
(BUILDMASK
 (LAMBDA (VAR VALIST)
 (COND ((GET VAR 'VALUESET)
        (PROG (**LST1 **VAL V VALUESET)
              (SETQ **LST1 VALIST)
              (SETQ VALUESET (GET VAR 'VALUESET))
         **LP (SETQ V (CAR (OR (LISTP **LST1) (GO **OUT))))
              (COND ((NOT (MEMBER V VALUESET))
                     (PRIN1 V)
                     (PRIN1 '$$" IS NOT A VALID VALUE FOR VARI")
                     (PRIN1 '$$"ABLE ")
                     (PRINT VAR)))
    **ITERATE (SETQ **LST1 (CDR **LST1))
              (GO **LP)
        **OUT (RETURN **VAL))
        (PROG (**LST1 **VAL VAL MASK I)
              (SETQ **LST1 (GET VAR 'VALUESET))
              (SETQ MASK 0)
              (SETQ I 0)
         **LP (SETQ VAL (CAR (OR (LISTP **LST1) (GO **OUT))))
              (COND ((NOT (MEMBER VAL VALIST)) (GO **ITERATE)))
              (SETQ MASK (MASKLOGOR MASK (LEFTSHIFT (MASKBIT VAR) I)))
    **ITERATE (SETQ **LST1 (CDR **LST1))
              (SETQ I (PLUS I 1))
              (GO **LP)
        **OUT (RETURN MASK)))
       (T
        (PROG (**LST1 **VAL VAL MASK)
              (SETQ **LST1 VALIST)
              (SETQ MASK 0)
         **LP (SETQ VAL (CAR (OR (LISTP **LST1) (GO **OUT))))
              (SETQ MASK
                    (MASKLOGOR MASK (LEFTSHIFT (MASKBIT VAR) VAL)))
    **ITERATE (SETQ **LST1 (CDR **LST1))
              (GO **LP)
        **OUT (RETURN MASK))))))
```

Function: BUILDLINEAR

Called By:
EXTENDAGAINSTL
NEGATELINEARSELECTOR
LINEARPRODUCT
LINEARPRODUCTDIFREL
ADDLINEAR

Function Listing:

```
(BUILDLINEAR (LAMBDA (VAR REL LOW HIGH) (LIST VAR REL LOW HIGH)))
```

Function: EARLIERVAR

Calls:
EARLIER
VAR

Function Listing:

```
(EARLIERVAR (LAMBDA (X Y) (EARLIER (VAR X) (VAR Y))))
```

Function: EVENTS

Called By:
READEVENTS
INCREMENTALGEN

Function Listing:

```
(EVENTS (LAMBDA (NAME) (GET NAME 'EVENTS)))
```

Function: FORMULA

Called By:
INCREMENTALGEN

Function Listing:

```
(FORMULA (LAMBDA (NAME) (GET NAME 'FORMULA)))
```

Function: GETMASK

Called By:
ADDNOMINAL

Function Listing:

```
(GETMASK (LAMBDA (VAR) (GET VAR 'MASK)))
```

Function: BUILDSTRUCTURE

Called By:
EXTENDAGAINSTSTRUCTURE
NEGATESTRUCTURESELECTOR
STRUCPRODUCT
ADDSTRUCTURED

Function Listing:

```
(BUILDSTRUCTURE (LAMBDA (VAR RELATION VAL) (LIST VAR RELATION VAL)))
```

Function: DOMAINTYPE

Called By:
READVL1

Function Listing:

```
(DOMAINTYPE (LAMBDA (VAR) (GET VAR 'DOMAINTYPE)))
```

Function: DOMINATE

Calls:
DOMINATE
PARENTS

Called By:
DOMINATE
MINLIST

Function Listing:

```
(DOMINATE
 (LAMBDA (V1 V2 VAR)
 (COND (V2 (OR (EQ V1 V2) (DOMINATE V1 (PARENTS V2 VAR) VAR))))))
```

Function: MASKBIT

Called By:
BUILDMASK

Function Listing:

```
(MASKBIT (LAMBDA (VAR) (GET VAR 'MASKBIT)))
```

Function: MASKEQUAL

Called By:
INCLUDESNOMINAL

Function Listing:

```
(MASKEQUAL (LAMBDA (X Y) (EQN X Y)))
```

Function: MASKLOGAND

Called By:
INCLUDESNOMINAL
EXTENDAGAINSTNOMINAL
RESTRICT
ADDNOMINAL

Function Listing:

```
(MASKLOGAND (LAMBDA (X Y Z) (LOGAND X Y (OR Z 7777777777777777777Q)))
                                                                )
```

Function: MASKLOGOR

Called By:
EXTENDAGAINSTNOMINAL
RESTRICT
NEGATECOMPLEX
ADDNOMINAL
BUILDMASK

Function Listing:

```
(MASKLOGOR (LAMBDA (X Y) (LOGOR X Y)))
```

Function: HIGH

Calls:
LOW

Called By:
EXTENDAGAINSTL
LINEARPRODUCT
LINEARPRODUCTDIFREL
PRINTLINEAR

Function Listing:

```
(HIGH (LAMBDA (X) (OR (CADDDR X) (LOW X))))
```

Function: LINEAR

Called By:
INCLUDES
EXTENDAGAINSTLINEAR
RESTRICT
SELECTOROFTRIVIALCOMPLEX
NEGATECOMPLEX
NEWCOMPLEXFROMLINEAR
NUMBEROFSELECTORS
ADDLINEAR
NEWCOMPLEXFROMSTRUCTURE
PPCOMP

Function Listing:

```
(LINEAR (LAMBDA (X) (CADR X)))
```

Function: LOW

Called By:
EXTENDAGAINSTLINEAR
HIGH
EXTENDAGAINSTL
LINEARPRODUCT
LINEARPRODUCTDIFREL
PRINTLINEAR

Function Listing:

```
(LOW (LAMBDA (X) (CADDR X)))
```

Function: NEWCOMPLEXFROMSTRUCTURE

Calls:
BUILDCOMP
NOMINAL
LINEAR
STRUCTURE

Function Listing:

```
(NEWCOMPLEXFROMSTRUCTURE
 (LAMBDA (COMP OLDSELECTOR NEWSELECTOR)
 (BUILDCOMP (NOMINAL COMP)
            (LINEAR COMP)
            (SORT
             (CONS NEWSELECTOR (REMOVE OLDSELECTOR (STRUCTURE COMP)))
             (FUNCTION EARLIERVAR)))))
```

Function: NOMINAL

Called By:
INCLUDES
EXTENDAGAINSTNOMINAL
RESTRICT
SELECTOROFTRIVIALCOMPLEX
READVARIABLES
NEGATECOMPLEX
NEWCOMPLEXFROMLINEAR
NUMBEROFSELECTORS
ADDNOMINAL
NEWCOMPLEXFROMSTRUCTURE
PPCOMP

Function Listing:

```
(NOMINAL (LAMBDA (X) (OR (NUMBERP (CAR X)) 7777777777777777777Q)))
```

Function: NOMINALP

Called By:
SELECTORTYPE

Function Listing:

```
(NOMINALP (LAMBDA (X) (NUMBERP X)))
```

Function: MASKLOGXOR

Called By:
EXTENDAGAINSTNOMINAL
NEGATECOMPLEX

Function Listing:

```
(MASKLOGXOR (LAMBDA (X Y) (LOGXOR X Y)))
```

Function: MAXBOUND

Called By:
EXTENDAGAINSTL

Function Listing:

```
(MAXBOUND (LAMBDA (VAR) (OR (GET VAR 'HIGH) 100000000)))
```

Function: NEGATION

Called By:
EXTENDAGAINSTNOMINAL
NEGATECOMPLEX
NUMBEROFSELECTORS
ADDNOMINAL

Function Listing:

```
(NEGATION (LAMBDA (X) (LOGXOR 7777777777777777777Q X)))
```

Function: SELECTOROFTRIVIALCOMPLEX

Calls:
NOMINAL
TOTALMASK
LINEAR
STRUCTURE

Called By:
MULTIPLY

Function Listing:

```
(SELECTOROFTRIVIALCOMPLEX
 (LAMBDA (COMP)
 (COND ((NOT (EQN (NOMINAL COMP) (TOTALMASK MASKLIST)))
        (NOMINAL COMP))
       ((CAR (OR (LINEAR COMP) (STRUCTURE COMP)))))))
```

Function: SETEVENTS

Called By:
READEVENTS
INCREMENTALGEN

Function Listing:

```
(SETEVENTS (LAMBDA (NAME VAL) (PUTPROP NAME 'EVENTS VAL)))
```

Function: SETFORMULA

Called By:
INCREMENTALGEN

Function Listing:

```
(SETFORMULA (LAMBDA (AT VAL) (PUTPROP AT 'FORMULA VAL)))
```

Function: NON-TRIVIAL

Called By:
INCLUDESSELECTORS

Function Listing:

```
(NON-TRIVIAL (LAMBDA (X) T))
```

Function: PARENT

Called By:
EXTENDAGAINSTSTRUCVAL
ALLPARENTS

Function Listing:

```
(PARENT (LAMBDA (VAL VAR) (CADR (GET VAL VAR))))
```

Function: PARENTS

Called By:
ABOVE
DOMINATE

Function Listing:

```
(PARENTS (LAMBDA (VAL VAR) (CADR (GET VAL VAR))))
```

Function: RELATION

Called By:
INBOUNDS
INCLUDESSTRUCSEL
NEGATELINEARSELECTOR
NEGATESTRUCTURESELECTOR
LINEARPRODUCT
PRINTLINEAR

Function Listing:

```
(RELATION (LAMBDA (X) (CADR X)))
```

Function: STRUCTURE

Called By:
INCLUDES
EXTENDAGAINSTSTRUCTURE
RESTRICT
SELECTOROFTRIVIALCOMPLEX
NEGATECOMPLEX
NEWCOMPLEXFROMLINEAR
NUMBEROFSELECTORS
ADDSTRUCTURED
NEWCOMPLEXFROMSTRUCTURE
PPCOMP

Function Listing:

```
(STRUCTURE (LAMBDA (X) (CADDR X)))
```

Function: TOTALMASK

Called By:
BUILDCOMP
SELECTOROFTRIVIALCOMPLEX
PRINTNOMINAL

Function Listing:

```
(TOTALMASK
 (LAMBDA (MASKLIST)
 (PROG (**LST1 **VAL MASK TOTALMASK)
       (SETQ **LST1 MASKLIST)
       (SETQ TOTALMASK 0)
  **LP (SETQ MASK (CAR (OR (LISTP **LST1) (GO **OUT))))
       (SETQ TOTALMASK (LOGOR MASK TOTALMASK))
 **ITERATE (SETQ **LST1 (CDR **LST1))
       (GO **LP)
 **OUT (RETURN TOTALMASK))))
```

Function: VALS

Called By:
EXTENDAGAINSTSTRUCTURE
EXTENDAGAINSTSTRUCVAL
NEGATELINEARSELECTOR
NEGATESTRUCTURESELECTOR
STRUCPRODUCT

Function Listing:

```
(VALS (LAMBDA (X) (CADDR X)))
```

Function: SETLINEAR

Called By:
ADDLINEAR

Function Listing:

```
(SETLINEAR
 (LAMBDA (COMPLEX LIN)
 (PROGN (RPLACA (CDR COMPLEX) LIN) COMPLEX)))
```

Function: SETNOMINAL

Called By:
ADDNOMINAL

Function Listing:

```
(SETNOMINAL (LAMBDA (COMPLEX NOM) (RPLACA COMPLEX NOM)))
```

Function: SETSTRUCTURED

Called By:
ADDSTRUCTURED

Function Listing:

```
(SETSTRUCTURED
 (LAMBDA (COMPLEX STRUC)
 (PROGN (RPLACA (CDDR COMPLEX) STRUC) COMPLEX)))
```

Function: SIBLINGS

Called By:
EXTENDAGAINSTSTRUCVAL

Function Listing:

```
(SIBLINGS (LAMBDA (VAL VAR) (CADDDR (GET VAL VAR))))
```

Appendix B-3

Input/Output Functions

Function: BITS

Called By:
PRINTNOMINALVAR

Function Listing:

```
(BITS
 (LAMBDA (MASK)
 (PROG (**VAL I)
       (SETQ I 1)
  **LP (COND ((NOT (ZEROP (LOGAND I MASK))) (GO **OUT)))
       (SETQ I (PLUS I I))
 **ITERATE (GO **LP)
 **OUT (RETURN
        (PROG (**VAL J **TEM1 **TEM2)
              (SETQ J I)
         **LP (COND ((ZEROP (LOGAND J MASK)) (GO **OUT)))
              (SETQ **TEM1 (PROG1 J (SETQ J (PLUS J J))))
              (COND (**TEM2
                     (RPLACD **TEM2 (SETQ **TEM2 (LIST **TEM1))))
                    (T (SETQ **VAL (SETQ **TEM2 (LIST **TEM1)))))
    **ITERATE (GO **LP)
        **OUT (RETURN **VAL))))))
```

Function: VAR

Called By:

INCLUDESSELECTORS
FINDSELECTOR
EXTENDAGAINSTLINEAR
EXTENDAGAINSTL
EXTENDAGAINSTSTRUCTURE
EXTENDAGAINSTSTRUCVAL
RESTRICT
SELECTORTYPE
NEGATELINEARSELECTOR
NEGATESTRUCTURESELECTOR
LINEARPRODUCT
LINEARPRODUCTDIFREL
NEWCOMPLEXFROMLINEAR
STRUCPRODUCT
EARLIERVAR
PRINTLINEAR

Function Listing:

```
(VAR (LAMBDA (X) (CAR X)))
```

Function: PPCOMP

Calls:
PRINTNOMINAL
NOMINAL
PRINTLINEAR
LINEAR
PRINTSTRUCTURE
STRUCTURE

Function Listing:

```
(PPCOMP
 (LAMBDA (COMP)
 (PROGN (PRINTNOMINAL (NOMINAL COMP))
        (PRINTLINEAR (LINEAR COMP))
        (PRINTSTRUCTURE (STRUCTURE COMP))
        (TERPRI))))
```

Function: PPCOMPS

Called By:
SHOWCOVERS
PPAQ
PPSTAR

Function Listing:

```
(PPCOMPS (LAMBDA (CL) (MAPC CL (FUNCTION PPCOMP))))
```

Function: PPSTAR

Calls:
PPCOMPS
STAR

Function Listing:

```
(PPSTAR (LAMBDA (X Y) (PPCOMPS (STAR X Y))))
```

Function: PN

Calls:

PRINTNOMINALVAR

Function Listing:

```
(PN
 (LAMBDA (NOMINAL)
 (PROG (**LST2 **LST1 **VAL MASK VARNAME)
       (SETQ **LST2 NOMINALVARS)
       (SETQ **LST1 MASKLIST)
  **LP (SETQ MASK (CAR (OR (LISTP **LST1) (GO **OUT))))
       (SETQ VARNAME (CAR (OR (LISTP **LST2) (GO **OUT))))
       (PRINTNOMINALVAR VAR (LOGAND MASK NOMINAL) MASK)
**ITERATE (SETQ **LST1 (CDR **LST1))
       (SETQ **LST2 (CDR **LST2))
       (GO **LP)
**OUT (RETURN **VAL))))
```

Function: PPAQ

Calls:

PPCOMPS
AQ

Function Listing:

```
(PPAQ
 (LAMBDA (ELIST FLIST)
 (PRIN1 '$$"POSITIVE EVENT SET IS:" ")
 (PPCOMPS ELIST)
 (TERPRI)
 (PRIN1 '$$"NEGATIVE EVENT SET IS:" ")
 (PPCOMPS FLIST)
 (TERPRI)
 (SETQ ELIST (AQ ELIST FLIST NIL))
 (PRIN1 '$$"COVER IS:" ")
 (PPCOMPS (CADR ELIST))
 (AND (NOT (ZEROP (CAR ELIST)))
      (PRIN1 '$$"EXCESS COMPLEXES NEEDED = ")
      (PRINT (CAR ELIST)))
 (PACK)))
```

AQLISP Users Guide
Appendix B-3

Function: PRINTNOMINAL

Calls:
TOTALMASK
PRINTNOMINALVAR

Called By:
PPCOMP

Function Listing:

```
(PRINTNOMINAL
 (LAMBDA (NOMINAL)
 (AND NOMINAL
      (NOT (EQN NOMINAL (TOTALMASK MASKLIST)))
      (PROG (**LST2 **LST1 **VAL MASK VARNAME)
            (SETQ **LST2 NOMINALVARS)
            (SETQ **LST1 MASKLIST)
       **LP (SETQ MASK (CAR (OR (LISTP **LST1) (GO **OUT))))
            (SETQ VARNAME (CAR (OR (LISTP **LST2) (GO **OUT))))
            (PRINTNOMINALVAR VARNAME (LOGAND MASK NOMINAL) MASK)
  **ITERATE (SETQ **LST1 (CDR **LST1))
            (SETQ **LST2 (CDR **LST2))
            (GO **LP)
      **OUT (RETURN **VAL)))))
```

Function: PRINTLINEAR

Calls:
VAR
RELATION
LOW
HIGH

Called By:
PPCOMP

Function Listing:

```
(PRINTLINEAR
 (LAMBDA (X)
 (AND X
      (MAPC X
            (FUNCTION
             (LAMBDA (SEL)
             (PROGN (PRIN1 _[)
                    (PRIN1 (VAR SEL))
                    (PRIN1 _ )
                    (PRIN1 (RELATION SEL))
                    (PRIN1 _ )
                    (COND ((NOT (EQN (LOW SEL) (HIGH SEL)))
                           (PRIN1 (LOW SEL))
                           (PRIN1 '$$"..")
                           (PRIN1 (HIGH SEL)))
                          (T (PRIN1 (LOW SEL))))
                    (PRIN1 _])))))))))
```

```
                    (SETQ I (PLUS I 1))
                    (GO **LP)
               **OUT (RETURN **VAL))))
          (PRIN1 _])))))
```

Function: PRINTSTRUCTURE

Called By:
PPCOMP

Function Listing:

```
(PRINTSTRUCTURE (LAMBDA (X) (AND X (PRIN1 X))))
```

Function: PRINTNOMINALVAR

Calls:
BITS

Called By:
PN
PRINTNOMINAL

Function Listing:

```
(PRINTNOMINALVAR
 (LAMBDA (VARNAME BITS MASK)
 (COND ((EQN BITS MASK) NIL)
       (T (PRIN1 _[)
          (PRIN1 VARNAME)
          (PRIN1 '$$" = ")
          (COND ((GET VARNAME 'VALUESET)
                 (PROG (**LST2 **LST1 **VAL B ALLBITS LOWORDER PRINTED
                        VARVALUE)
                       (SETQ **LST2 (GET VARNAME 'VALUESET))
                       (SETQ **LST1 (BITS MASK))
                       (SETQ ALLBITS (BITS MASK))
                       (SETQ LOWORDER (CAR (BITS MASK)))
                  **LP (SETQ B (CAR (OR (LISTP **LST1) (GO **OUT))))
                       (SETQ VARVALUE
                             (CAR (OR (LISTP **LST2) (GO **OUT))))
                       (COND ((ZEROP (LOGAND B BITS)) (GO **ITERATE)))
                       (AND PRINTED (PRIN1 _,))
                       (PRIN1 VARVALUE)
                       (SETQ PRINTED (OR PRINTED T))
           **ITERATE (SETQ **LST1 (CDR **LST1))
                       (SETQ **LST2 (CDR **LST2))
                       (GO **LP)
              **OUT (RETURN **VAL)))
                (T
                 (PROG (**LST1 **VAL B ALLBITS LOWORDER PRINTED I)
                       (SETQ **LST1 (BITS MASK))
                       (SETQ ALLBITS (BITS MASK))
                       (SETQ LOWORDER (CAR (BITS MASK)))
                       (SETQ I 0)
                  **LP (SETQ B (CAR (OR (LISTP **LST1) (GO **OUT))))
                       (COND ((ZEROP (LOGAND B BITS)) (GO **ITERATE)))
                       (AND PRINTED (PRIN1 _,))
                       (PRIN1 I)
                       (SETQ PRINTED (OR PRINTED T))
           **ITERATE (SETQ **LST1 (CDR **LST1))
```

Function: READCOMPLEX

Calls:
READVL1

Called By:
READEVENTS

Function Listing:

```
(READCOMPLEX
 (LAMBDA NIL
 (READVL1
  (PROG (**VAL NEWSEL **TEM1 **TEM2)
   **LP (COND ((EQ (SETQ NEWSEL (READ)) ';) (GO **OUT)))
        (SETQ **TEM1 NEWSEL)
        (COND (**TEM2 (RPLACD **TEM2 (SETQ **TEM2 (LIST **TEM1))))
              (T (SETQ **VAL (SETQ **TEM2 (LIST **TEM1)))))
 **ITERATE (GO **LP)
  **OUT (RETURN **VAL)))))
```

Function: SHOWCOVERS

Calls:
PPCOMPS

Called By:
AQVAL

Function Listing:

```
(SHOWCOVERS
 (LAMBDA (CS CLASSNAMES)
 (PROGN (TERPRI)
        (TERPRI)
        (PRIN1 '$$"THE COVERS ARE:")
        (TERPRI)
        (COND (CLASSNAMES
               (PROG (**LST2 **LST1 **VAL C CLASSNAME)
                     (SETQ **LST2 CLASSNAMES)
                     (SETQ **LST1 CS)
                **LP (SETQ C (CAR (OR (LISTP **LST1) (GO **OUT))))
                     (SETQ CLASSNAME
                           (CAR (OR (LISTP **LST2) (GO **OUT))))
                     (PRINZ (LIST '$$"COVER OF CLASS: " CLASSNAME))
                     (TERPRI)
                     (PPCOMPS (CADR C))
                     (TERPRI)
                     (TERPRI)
          **ITERATE (SETQ **LST1 (CDR **LST1))
                     (SETQ **LST2 (CDR **LST2))
                     (GO **LP)
              **OUT (RETURN **VAL)))
              (T
               (PROG (**LST1 **VAL C I)
                     (SETQ **LST1 CS)
                     (SETQ I 1)
                **LP (SETQ C (CAR (OR (LISTP **LST1) (GO **OUT))))
                     (PRINZ (LIST '$$"COVER OF CLASS #" I))
                     (TERPRI)
                     (PPCOMPS (CADR C))
                     (TERPRI)
                     (TERPRI)
           **ITERATE (SETQ **LST1 (CDR **LST1))
                     (SETQ I (PLUS I 1))
                     (GO **LP)
                **OUT (RETURN **VAL)))))))
```

Function: READVARIABLES

Calls:
SETUPMASKLIST
NOMINAL

Called By:
AQVAL

Function Listing:

```
(READVARIABLES
 (LAMBDA NIL
 (PROGN (PRIN1 '$$"DEFINE VARIABLES FOR YOUR PROB")
        (PRIN1 '$$"LEM: (END WITH '*')")
        (TERPRI)
        (PROG (**VAL VARS VALUESET LOWVALUE HIGHVALUE DOMAINTYPE)
              (SETQ NOMINALVARS
                    (SETQ LINEARVARS (SETQ STRUCTUREDVARS NIL)))
         **LP (COND ((NOT
                      (SELECT
                       (CAR
                        (SETQ VARS
                              (PROGN (TERPRI)
                                     (PRIN1 '$$"VARIABLE NAME(S): ")
                                     (READLINE))))
                       ('* NIL)
                       VARS))
                     (SETQ MASKLIST (SETUPMASKLIST NOMINALVARS))
                     (RETURN **VAL)))
                    (SETQ DOMAINTYPE
                          (ASKUSER NIL
                                   'N
                                   '($$"DOMAIN TYPE: ")
                                   '(NOMINAL LINEAR STRUCTURED)
                                   T))
                    (SELECT DOMAINTYPE
                            ('NOMINAL
                             (PRIN1 '$$"WHAT ARE THE PERMISSIBLE VALUE")
                             (PRIN1 '$$"S: ")
                             (TERPRI)
                             (PRIN1 '$$"(GIVE VALUES, SEPARATED BY SPA")
                             (PRIN1 '$$"CES OR COMMAS, END WITH '*')")
                             (SETQ VALUESET (READLINE))
                             (MAPC VARS
                                   (FUNCTION
                                    (LAMBDA (VAR)
```

Function: READEVENTS

Calls:
READCOMPLEX
SETEVENTS
EVENTS

Called By:
AQVAL

Function Listing:

```
(READEVENTS
 (LAMBDA NIL
 (PROGN (TERPRI)
        (TERPRI)
        (PRINT '$$"ENTER EVENTS AND CLASSES: ")
        (PROG (**VAL PREVCLASS CLASS CLASSLIST)
         **LP (COND ((NOT
                      (PROGN (PRIN1 '$$"EVENT: ")
                             (SETQ COMP (READCOMPLEX))))
                     (GO **OUT)))
              (SETQ PREVCLASS
                    (OR
                     (PROGN (PRIN1 '$$"CLASS: ")
                            (SELECT (SETQ CLASS (READ)) (NIL) CLASS))
                     PREVCLASS))
              (SETEVENTS PREVCLASS
                         (NCONC (EVENTS PREVCLASS) (LIST COMP)))
              (SETQ CLASSLIST (UNION (LIST PREVCLASS) CLASSLIST))
   **ITERATE (GO **LP)
       **OUT (RETURN (SORT CLASSLIST NIL))))))
```

AQLISP Users Guide
Appendix B-3

Function: SETUPMASKLIST

Called By:
READVARIABLES

Function Listing:

```
(SETUPMASKLIST
 (LAMBDA (NOMINALVARS)
 (PROG (**LST1 **VAL VAR SHIFT **TEM1 **TEM2)
       (SETQ **LST1 NOMINALVARS)
       (SETQ SHIFT 0)
  **LP (SETQ VAR (CAR (OR (LISTP **LST1) (GO **OUT))))
       (SETQ **TEM1
             (PROG1
              (PROG (**LST1 **VAL VAL MASK)
                    (SETQ **LST1 (CDR (GET VAR 'VALUESET)))
                    (SETQ MASK 1)
                **LP (SETQ VAL (CAR (OR (LISTP **LST1) (GO **OUT))))
                    (SETQ MASK (LOGOR 1 (LEFTSHIFT MASK 1)))
          **ITERATE (SETQ **LST1 (CDR **LST1))
                    (GO **LP)
              **OUT (PUTPROP VAR 'MASKBIT (LEFTSHIFT 1 SHIFT))
                    (RETURN (PUTPROP VAR 'MASK (LEFTSHIFT MASK SHIFT))
                                                                  ))
              (SETQ SHIFT (PLUS SHIFT (LENGTH (GET VAR 'VALUESET))))))
       (COND (**TEM2 (RPLACD **TEM2 (SETQ **TEM2 (LIST **TEM1))))
             (T (SETQ **VAL (SETQ **TEM2 (LIST **TEM1)))))
**ITERATE (SETQ **LST1 (CDR **LST1))
       (GO **LP)
**OUT (RETURN **VAL))))
```

```
                              (PROGN
                               (PUTPROP VAR 'DOMAINTYPE 'NOMINAL)
                               (PUTPROP VAR 'VALUESET VALUESET)
                               (SETQ NOMINALVARS
                                     (NCONC NOMINALVARS (LIST VAR)))))
                                                                  )))
                         ('LINEAR
                          (PRIN1 '$$"WHAT IS THE RANGE OF PERMISSIB")
                          (PRIN1 '$$"LE VALUES: ")
                          (TERPRI)
                          (PRIN1 '$$"(TYPE AS 'LOWVALUE,HIGHVALUE'")
                          (PRIN1 '$$") ")
                          (SETQ LOWVALUE (READ))
                          (SETQ HIGHVALUE (READ))
                          (MAPC VARS
                                (FUNCTION
                                 (LAMBDA (VAR)
                                 (PROGN (PUTPROP VAR 'DOMAINTYPE 'LINEAR)
                                        (PUTPROP VAR 'LOW LOWVALUE)
                                        (PUTPROP VAR 'HIGH HIGHVALUE)
                                        (SETQ LINEARVARS
                                           (NCONC LINEARVARS (LIST VAR)))
                                                                  )))))
                        ('STRUCTURED
                         (PRIN1 '$$"SORRY, CAN'T DO THAT YET")
                         (RPRI))
                        (SHOULDNT))
**ITERATE (GO **LP)))))
```