78-5

78-5

Description of Inductive Program INDUCE 1.1

by

T. Dietterich

Internal Report

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

October 1978

TABLE OF CONTENTS

1. Introduction

This document is in support of the paper [1] to provide further details of the implementation of the program INDUCE_1. This program accepts an environment description, a set of VL decision rules, and a set of parameters. The program produces a set of generalizations of the input decision rules. The basic algorithms and input syntax are given in chapter 5 of the paper [1] so will not be repeated in full here. In the following pages, the actual commands necessary to use the program are given. Chapter 2 contains a description of the data structures used in the program. The reader is referred to the program listing for more detailed structure. In chapter 3, the various I/O files are described. Chapter 4 gives a brief outline of the purpose of each procedure and its relation to other procedures in the program. The appendix provides a listing of the program for the CYBER machine and a BOSS editor macro for converting the CYBER version to a DEC-10 version.

1.1 High level commands

The following single letter comands can be entered into the program to perform various functions:

M (modify rule base) - This command is used to enter rules into the program or delete rules from memory. Following the M command, the user may enter (A) to add a new rule, (D) to delete an existing rule, or anything else to return to the main level without doing anything. After an A is entered, the system expects a VL rule in correct syntax terminated with a period (.). Since there is no online error correction, this is usually done by placing all rules in a local file (CFILE) with the commands (M and A) interspersed. After the rule has been entered, the program returns to the high level command mode. If a (D) is entered, the program proceeds through the list of all rules asking at each stage whether to delete the rule.

The user may enter Y, N, or Q to delete the rule and move to the next rule, to keep the rule and move to the next, or return to the command level.

Example:

[SHAPE(X1)=1][P(X1,X2)=2] => [D=2].

H (get help) - Enter this command to obtain a brief explanation of the high level commands and a detailed explanation of one such command by entering 'H X' where x is one of the letters corresponding to a high level command.

R (enter restrictions) - Enter R (carriage return) followed by the restrictions which are to be added to each of the rules entered. Each argument in the right hand side must appear in the left hand side and the left hand side must form a connected graph structure. As with all rules, the restriction rule must end with a period.

Example

[ONTOP(P1,P2)][ONTOP(P2,P3)] => [ONTOP(P1,P3)].

E (enter domain generalization structures) - Enter tree structure for such domains. These must be entered in order from lowest level generalization to highest level generalization. For VL applications, this should be done after a V command has been entered since the V command initializes the symbol table for the special VL mode.

Example:

[SHAPE=2,4] => [SHAPE=10].
[SHAPE=0,1,3,5] => [SHAPE=11].
[SHAPE=6,7,8,9] => [SHAPE=12].
[SHAPE=10,11] => [SHAPE=13].

L,S (Enter EXTNTY (L) and EQUIV (S) type predicates). Just enter the one letter command to add either type of generated predicate. (There is currently no way of removing such a predicate from a structure except by re-running the program).

A (Enter an arithmetic derived descriptor) - Enter the derivation rule for an arithmetic derived descriptor.

Example:

GIRTH(X1)=LENGTH(X1)+WIDTH(X1).

Restrictions: the dummy variables of the function on the left hand side must appear on the right hand side of the equation. The arithmetic expression is written in standard algebraic form. The operators which may be used are: + (addition), - (subtraction), - (unary minus), * (multiplication), / (integer division--remainder discarded), and % (integer modulus). *, /, and % are evaluated before - and +. Integer constants may also appear in the expression. The right hand side must contain at least one function or predicate. All functions and predicates are assumed to have interval domains. If more than one value appears in the reference of a function when the expression is to be evaluated, the smallest value is used.

The right hand side must form a connected graph structure as well. A connecting predicate can be multiplied to the original expression to accomplish this since predicates have values of 1 when true.

Bug: This command will not work correctly if there are two or more occurrences of exactly the same function (with the same dummy variables) on the right hand side.

N (Add arithmetic derived descriptors to the rule base) - The separation of the A and N commands is included to permit users to enter the rules and the arithmetic descriptors in any order and then to apply the arithmetic descriptors when they are desired (after all of the rules have been read in). The N command causes all previously entered arithmetic derived descriptors (since the most recent N command) to be processed and added to all rules in the rule base where they are appropriate.

X (Enter a logical derived descriptor and substitute it into the rule base)

- Logical derived descriptors are handled by two separate commands. The R command permits the user to enter a logical derived descriptor which is to be added to

each rule for which the premise is true. The X command permits the user to enter a logical derived descriptor which is to be substituted (exchanged) for its premise in each rule in which the premise is true. (The premise is the left hand side of the rule.)

Example:

[BIG(PART1)][BOX(PART1)] => [BIGBOX(PART1)].

This example command will substitute [BIGBOX(PARTN)] for every conjunction of BIG(PARTN) and BOX(PARTN) where PARTN is any given PART dummy variable.

Each dummy on the right hand side must appear on the left hand side. The right hand side must be a single selector. The left hand side must form a connected graph structure.

C (Cover a set of formulas) - Enter the number of the associated decision after the C command. Be sure to set any trace information using the appropriate parameters before entering the C command.

V (VL_1 mode) - This mode bypasses the VL_2 type structure creation and accepts VL_1 events from the file VL1EVE.

After entering V, the program asks for the number of variables which are to be used. Enter this number (it should be 1 less than the number of entries in each line of the VL1EVE file because of the class number in the file). Then, the user is asked to enter another command (E, C, Q, or P). Enter E and then a domain generalization structure for that type of domain, P to change parameters (AQMAXSTAR, LQST, AQCRIT, AQTOLERANCE, or enter VCOST or VTYPE, the latter may be necessary for interval type variables), C to cover a set of events, or Q to return to the high level commands. All of the E and P parameters may be included in CFILE. When C is entered, the program requests the number of the class of events to be covered

and then the number(s) of the class(es) against which the cover should be made. To cover against all other classes, enter -1 instead of a list of all other classes. (This is useful for intersecting type covers.) All specifications may be placed in CFILE.

P (Parameters) - This places the user in a parameter examination and modification mode. To get an explanation of each parameter on-line, enter
HELP <parameter name> or HELP
the latter to get a list of parameters. See the EXPLAIN file for a list of all the parameters and explanations. No checking is done to see if parameter values are in the right range. A missing value is interpreted as the value 0. Most parameters require the parameter name followed by the value. Parameters which may be true or false are set to true by entering the parameter name (e.g. LQST) and are set to false by entering the parameter followed by F (e.g. LQST F). Trace and stop parameters are turned on one at a time by entering TRACE or STP and then the associated number. They are turned off by entering the negative of the number (e.g. TRACE 3 turns on trace 3, STP -6 turns off the program stop at trace level 6).

Functions such as VCOST and VTYPE must have the associated descriptor name in parentheses following the parameter name (e.g. VTYPE(SHAPE)=2 sets the domain of SHAPE to type interval.) All VL type variables have descriptor names X1, X2, ... Xn (so VCOST(X1)=-2 sets the cost of the variable X1 to -2). After all parameters have been set, entering QUIT returns to the previous command. In order to examine the parameters, enter PARA and enter PRINT D to examine the domains of all functions in the symbol table. PARA will give the type and cost of all functions for which the two characteristics VTYPE and VCOST are not the default values (type nominal and cost of 0).

Q (Quit) - Halts the program.

D (Dump) - This command, used during debugging, dumps the rule base graph structure and the symbol table on file OUTPUT.

1.2 Parameters

This section describes the parameters which can be modified after entering the command P above and the commands required to inspect the parameters in the running version of the program. The parameters and their meaning are as follows, default values are in parentheses:

TRACE - This parameter may have a set of values in the interval [1..10]. Each value relates to a trace feature of the program. The values currently meaningful are:

1 - Print all of the c-formulas in each untrimmed and each trimmed partial star to examine the process of consistent formula generation and trimming.

2 - Print all the consistent formulas both before the AQ7 generalization and after this generalization.

3 - Print the best AQ formula; i.e. select the best formula from the output of trace 2.

4 - Print the input events to the AQ7 procedure and the variable association between the VL c-structure and the VL variables.

5 - Print the output from the VL AQ7 procedure.

6 - Print the selected meta functions in a table.

7 - Print the LQST2 process during characteristic generalization.

8 - Not used.

9 - Print all generalizations of an event (i.e. the complete set of alternative generalizations which the program has calculated for one event from trace 10). This is the same as the list which comes from trace 2 without the input formulas to AQ7.

10- Print the event (c-formula) which is to be covered from P1.

To turn on (off) any trace feature, enter

TRACE i (or TRACE -i)

where i is the number of the trace feature to be turned on (off).

STP - This parameter may also have a set of values in the range [1..10]. Each value corresponds to one trace feature defined above. If STP contains a value of a trace feature and the particular trace feature is set, then the program pauses at the point where the trace information is printed and will provide an explanation of the situation or allow the user to modify parameters. STP may be turned on and off in the same way as TRACE, i.e.

STP i (or STP -i)

AQCUTF1 (20) - This is a limit on the number of c-formulas examined using the AQ cost function 3.

AQMAXSTAR(2) - This is the AQ maxstar parameter (the number of complexes retained in a partial star in the AQ7 procedure).

AQCRIT(-1,2) - The criteria list of cost functions to be applied in the AQ procedure. There are six cost functions available:

1 - Measure the number of events covered by a complex which are not covered by any previously generated L_q complex.

2 - Measure the number of selectors whose reference is not equal to *.

3 - Measure the number of c-formulas which are actually covered by a complex. This is more time consuming than 1 but may give better results.

4 - Sum the costs of all variables in a complex in selectors whose reference is not equal to *.

5 - Measure the number of events in the set F1 which are covered by the complex.

6 - Find the number of events in the set 2 (F0).

To specify a cost criterion, enter

AQCRIT(I)=J

where j is the number of the criterion (if negative, then the cost is computed as the negative of the value determined by the criterion), and i is the order of application of the criterion.

AQTOLERANCE(0) - This is the tolerance associated with each criterion specified in AQCRIT above. AQTOLERANCE(I) is the tolerance associated with criterion AQCRIT(I). The tolerance can be an absolute tolerance (if it is greater than 1) or a relative tolerance (if it is less than 1). The tolerance is always specified in hundreths, e.g.:

AQTOLERANCE(2)=200

results in a an absolute tolerance of 2 for the criterion applied second.

AQNF(2) - The number of criteria which are to be applied to the complexes.

LQST(TRUE) - If LQST is set, then the resulting complexes from the AQ7 procedure are stripped to only the necessary values in the reference. To turn off this feature, enter

LQST F

VIMAXSTAR(2) - The maximum number of formulas retained in a partial star.

VLCRIT(3,-1,2) - The criteria list which is to be used for trimming VL formulas. There are five criteria available:

1 - Count the number of c-formulas which are covered by this formula.

2 - Count the number of selectors in the formula.

3 - Count the number formulas of the set FO which intersect with this formula.

4 - Sum the total cost of all references in all selectors of the formula with reference not equal to *.

5 - Sum the cost of all dummy variables used in the function and predicate selectors of the formulas. This uses the cost of a specific dummy variable (e.g. X1) as originally entered (not as dynamically reassigned by the program). It uses the DPNO field.

This parameter is specified in the same way as AQCRIT above.

VLTOLERANCE(.3,0,0) - The tolerance associated with each VLCRIT specified above. See AQTOLERANCE above for details about how to enter values for this parameter.

VLNF(3) - The number of VL criteria to apply when trimming a list of formulas.

NCONSIST(2) - The number of consistent alternative generalizations which the program is to produce.

ALTER(2) - The number of alternative new formulas which are produced from one formula when creating a new partial star from an old one.

VCOST(0) - The cost of each function in the system. All VL1 variables when running in VL mode are labelled X1,X2,....,XN. To enter a cost, type:

VCOST(<fn-name>)=i

where <fn-name> is the name of a function which has been in a decision rule which is currently in the program, and i is the cost of the function. Some examples:

VCOST(SHAPE) = 2 or VCOST(X4)= 1

VTYPE(1) - This is the structure of each domain:

1 - nominal
2 - interval
3 - tree structured.

The type 3 is set automatically when the F command is entered. To make a function domain into an interval type, enter:

VTYPE(SHAPE) = 2

METATRIM(3) - This specifies the number of different meta functions which are to be selected by the program to be used in descriptions. This value should be less than GSIZE. If it is 0, then no meta-functions are generated.

DESCTYPE (DISCRIMINANT) - This specifies the type of description which the program is to generate. DESCTYPE DISCRIMINANT causes the program to generate the most general description which discriminates events of set F1 from events of set F0. DESCTYPE CHARACTERISTIC causes the program to generate the most specific description which is shared by all events in set F1. F0 must be empty for this to work properly. Thus, only one set of events should be supplied to the program for a characteristic description.

For characteristic descriptions, the parameter MINCOVER must be set.

MINCOVER (100) - This specifies the minimum percentage of rules in P1 that a description must cover in order to be considered as a characteristic description. During the rule growing process, each rule is grown (by adding additional selectors) until it fails to cover MINCOVER% of the rules in P1. At that time, it is placed on the MQ star. NCONSIST such MQ rules must be found before the growing algorithm terminates. Thus, if MINCOVER=100, several, fairly trivial rules will be found. If MINCOVER=50, some interesting rules will be found (but this will use more cpu time) but these rules may not cover all of P1.

PRINT X - This allows the user to examine certain tables in the program. X may be one of P, R, D, M and the system will respond by listing:

P - The set of input decision rules

R - The set of input restrictions

D - The domain table

M - The currently selected meta-functions.

PARAMETERS - This lists the current parameter values in a table.

QUICK - This turns off all trace values

BRIEF - This sets the trace options 3,9,10 and stop option 10.

DETAIL - This sets all traces.

EXPLAIN - This sets all traces and all stop options.

HELP - This allows the user to obtain an explanation on-line of the function of any of the parameters and a list of all parameters accepted under the P high level command.

QUIT - This returns the user to what ever he was doing before entering the parameter modification section.

2.0 Data Structures

2.1 Constants

Some constants in the program control the sizes of many structures which may be sensitive to the current problem characteristics. These constants may be increased (to allow larger data structures) or decreased (to permit more copies of a data structure in memory at one time). The constants and their use appear below (suggested values are in parentheses).

SYMSZE(36) - is the size of the symbol table. It can be estimated by finding the sum of the number of functions, predicates, and distinct variables plus the number of groups of variables plus 2 (for meta functions #PT and FORALL) plus 2 times the number of binary predicates (for MST-, LST- type predicates). In VL mode, SYMSZE is the number of VL variables plus 1.

NDES(15) - is the size of the DSTRUC table. One row is required in this table for each internal node in each generalization structure (i.e. one row for each rule which is input with the E command.)

GSIZE(30) - specifies the size of all graph structures in the program and the number of VL type variables which are allowed in the program. This number being too small is probably the cause of an 'array index out of bounds' message and may be remedied by increasing the parameter. Its value can be estimated by finding the sum of the number of selectors in the longest rule which must be stored plus the number of variables in the rule plus 1 (not including meta selectors). An estimate which is too large will use up memory very quickly and cause a message 'runtime stack overflow' therefore, the parameter should be approximated rather closely.

MNVAL(15) - is the maximum value in a set of values. A set of values (VALTP) is used in several places (GRAPH, CPX, DSTRUC) in the program. Each set is allowed to contain values from 0 to MNVAL. There is a maximum value of this parameter determined by the architecture of the machine (CDC is 58, DEC is about 30).

MLNK(18) - is the number of links to any node of a graph structure. This may be estimated by finding the maximum number of times that a particular variable occurs in a rule and using either this figure or the larger number of arguments of any one function, which ever is largest.

MLNK must be one larger than either of these numbers since links are stored as an array of numbers which terminates with a 0 value.

MRULE(50) - is the maximum number of rules in either F1 or F0.

MAXASTACK(20) - is the maximum number of entries in an arithmetic expression stack. There is one entry on the stack for each function and value in the expression and one entry on the stack for each operator. There is no compiler or system limit to this parameter.

2.2 Parse table (PT)

The parse table consists of a data structure which represents the productions in the VL grammar (RHS and CONT) along with information about which semantic routines are invoked with the recognition of one non-terminal in the grammar (SRULE). The array RHS contains a row for each alternative in each production where each element in a row is a positive or negative integer or zero. If the number is positive, it represents a token in the input (it is either the machine representation of a character or 1 - a function symbol, 2 - a variable, or 3 - a number). If the entry of RHS is negative, it represents a non-terminal whose definition is found beginning in the row corresponding to the absolute value of the entry (e.g. -3 represents the non-terminal beginning in row 3 of the table). A zero value signifies the end of the alternative. The boolean array CONT indicates whether a row of RHS is a continuation of a previous row in a production (value true) or the first alternative of a production (value false). Finally, the array SRULE contains a number indicating the semantic rule (element in a case statement in the procedure PROCESS) which is to be applied if the production in the corresponding row of the table is matched.

Example: (see file TABLES for the complete input grammar)

```
<VLRULE>        $$= <NUMBER> <RULE> ^ <RULE>
<RULE>          $$= <CONDITION> => <SELECTOR>
<CONDITION>     $$= <SELECTOR> <CONDITION> ^ <SELECTOR>
<SELECTOR>      $$= [ <VARIABLE> = <REF> ] ^
                    [ <FN-SYM> ( <ALIST> )= <REF> ]
```

Parse Table in the program: (The actual table in the program contains numbers instead of characters)

ROW	SRULE	CONT	RHS
1	1	F	3 -3
2	2	T	-3 0
3	3	F	-4 = > -6 0
4	4	F	-6 -4 0
5	5	T	-6 0
6	14	F	[-19 = -10] 0
7	7	T	[-21 (-14) = -10] 0

2.2 Symbol Table (SYMTAB)

The symbol table is a table with an entry for each function, variable, and symbolic value in the VL decision rules. One entry (NELT) specifies the number of rows which are actually used. The first two rows always contain the information for the meta functions #PT and FORALL. The columns contain:

NAME - the character string representing the name of the entry

PNO - the function number associated with the entry (normally this just points to the row which contains the entry).

DPNO - for variables, this points to (contains the index of) the row which contains the domain definition of the particular entry (e.g. the row with X4 would point to the row containing the entry for X). For functions, this is the head of a linked list linking, in order, the symbolic names for the reference values of this function.

NARG - the number of arguments of a function.

VTYPE - domain structure (1-nominal, 2-interval, 3-tree structured).

VCOST - variable cost used in cost functions 4 and 5 and selection of alternative selectors (ALTER parameter) in the procedure NEWGP.

EVAL - maximum value in complete domain (including all nodes in the generalization structure).

NVAL - number of leaves of tree structure domain. (EVAL = NVAL for non tree structure domains).

MVAL - minimum value in the domain.

Example: NELT=7

NAME	DPNO	PNO	NARG	VTYPE	VCOST	EVAL	NVAL	MVAL
FORALL	0	1	0	1	0	1	1	1
#PT	0	2	0	2	0	6	6	0
SHAPE	0	3	1	3	-1	8	6	1
X	4	4	0	1	0	15	15	0
X1	4	5	0	1	0	15	15	0
X2	4	6	0	1	0	15	15	0
P	0	7	2	1	0	1	1	1

2.4 Domain Structures (DSTRUC)

The generalization structures of each tree structured domain are stored in this record. Again, NELE specifies the number of rows in the table which are used. PREM is a set of all descendents of the node in CONS for the domain of the function which is defined in the row PNO of the symbol table.

Example:

[SHAPE=1,2,3] => [SHAPE=7].

[SHAPE=0,5,6] => [SHAPE=8].

PREM	CONS	PNO
1,2,3	7	3
0,5,6	8	3

2.5 Meta selector Table (MSTR)

This table records the meaning of meta selectors which are used in the formulas. The values of the selector themselves are stored in a structure referenced by MSEL in the GRAPH record. The table contains two integers (METATRIM and NMST) the latter indicates the number of current entries in the table. Elements of the table are accessed indirectly through the array PTR to facilitate sorting of the array with a minimum amount of effort.

(e.g. the third element logically in the array PNO is the element PNO[PTR[3]]). Elements are sorted in descending order using PTR as an index according to the values of F1COV (primary field) and -F0COV (the secondary field). The columns are interpreted:

SYMPTR - is the index in the symbol table of the name of the meta function (e.g. a pointer to either FORALL or #PT).

VARPTR - is the index into the symbol table of the dummy variable associated with the unary function from which the metaselector is derived. (e.g. for [shape(X1)=...] VARPTR points to X).

PNO - is the index in the symbol table of the referee associated with the particular meta function (e.g. a pointer to SHAPE in the symbol table for a function which counts the number of occurrences of a selector of the form [shape(x1)= ...]).

VAL - is the set containing the reference of the function associated with PNO (e.g. the reference in a selector [SHAPE(X1)=2,3]).

PTR - is the location in PNO, SYMPTR etc. of the information for each selected meta selector in the order of preference (e.g. information for MS2 would be found in PNO[PTR[2]], SYMPTR[PTR[2]] etc).

F1COV - the maximum number of formulas in F1 covered by one value of this meta function.

F0COV - is the number of formulas of F0 covered by the meta function with the value found in F1COV.

Example: (N=ST=3)

PNO	VAL	SYMPTR	VARPTR	PTR	F1COV	F0COV
3	1	1	4	2	3	0
3	0	2	4	1	4	0
3	1	2	4	3	3	2

with the three meta functions:

MS1 = [# XS SHAPE 0=...]

MS2 = [ALL XS SHAPE 1]

MS3 = [# XS SHAPE 1=...]

2.6 Formula for Graph Structure (GRAPH)

This is the structure used to store each formula. It is composed of 4 parts, the single parameters (COEF, RNO, COST, ESET, NXTN), a pointer to a set of meta selectors (MSEL), and information about each node and the links between nodes. Each node has a number (the subscript value of each array below) which is used in the LNK array to refer to any node in the graph so that for example, VAL[3] is the value set associated with the node number 3.

COEF - not used

RNO - the unique rule number associated with the graph.

FP - a flag which is used in absorption and the COVER routine.

COST - the cost of the formula (COST[I] is the value associated with cost criterion number I).

ESET - the decision value associated with this rule

NXTN - the pointer to the next graph structure in a list or set of such structures.

NNEG - not used.

MSEL - a pointer to the meta selectors associated with the graph. The metaselectors are stored in an AQ7 complex corresponding to the MST.

VBL - if true, then the node is a variable, otherwise, it is a selector node.

ORDIRR - if true, then the order of arguments is irrelevant (i.e. all connecting edges are unlabeled). In general, dummy variables and equivalence-type predicates have ORDIRR=TRUE and all other functions and predicates do not.

VAL - the set of values associated with the node (this may be a subrange corresponding to [X1=3..6] for example).

CCUNT - this is used in NEWGP and AQSET when generating alternative generalizations. In general, a non-zero value indicates that a node is in the graph.

ASSGN - records assignments between nodes of two different graphs in SUBG1 when a 1-1 correspondence between nodes of two graphs is determined.

PNO - a pointer to the domain definition for the function in the symbol table. Points to the dummy variable family name (e.g. PART instead of PART1).

DUMNUM - is used in VLINT and PGRAPH to distinguish between two variables with the same domains (e.g. x1 and x2).

DPNO - A pointer to the domain definition of the dummy variable itself. It points to, e.g., PART1 rather than PART (unlike PNO). It is used by VCOST function 5 to derive the correct cost.

LNK - contain the links between nodes. Edges are not given an explicit direction, instead, certain routines infer the direction of an edge by the types of node at each end of the edge. All nodes which are connected are doubly linked; if incomming edges are labeled, these labels are indicated by the location in the link array (LNK) for the node.

Example

For the expression: [P(X1,X2)][SHAPE(X1)=2],

the link structure is

ROW	FUNCTION	LINKS
1	X2	3 0
2	X1	3 4 0
3	P	2 1 0
4	SHAPE	2 0

A partial example using the symbol table above is:

[SHAPE(X1)=1][P(X1,X2)][MS2=2]

NODE	PNO	VAL	VBL	ORDIRR	LNK
1	4	0..15	TRUE	TRUE	2 3 0
2	3	1	FALSE	FALSE	1 0
3	7	1	FALSE	FALSE	1 4 0
4	4	0.15	TRUE	TRUE	3 C

MSEL^: [MS1=*][MS2=2][MS3=*]

2.7 VL Complex Storage (CPX)

This structure is a simple list of references (CVAL) in bit positional notation along with certain flags (FP and FQ), a link to the next such structure in a set (NXTC) and the cost of the complex (COST). The interpretation of each variable is found in the symbol table through the index SLOC in AQPARM (e.g. the set contained in CVAL[3] is the reference of the variable in row SLOC[3] of the symbol table).

2.8 AQ7 Parameters (AQPARM)

The structure contains several parameters relevant to the AQ7 procedure.

NVAR - the number of variables for the run.

CSTF - the list of cost functions in the order of application.

TOLER - the tolerance asociated with each cost function (TOLER[3] is the tolerance of the cost function which is applied third -- i.e. CSTF[3]).

NF - the number of cost functions to apply

FREEC - a pointer to a list of free complex storage structures (CPX's)

SLOC - the location in the symbol table of the domain definition for each VL type selector in CVAL.

CUTF1 - a parameter which limits the number of formulas examined with AQCRIT of 3.

LOST - if true, then VL complexes are stripped.

MAXSTARAQ - the maximum size of a partial star in AQ7

2.9 VL Parameters (PARM)

This structure contains parameters relevant to the VL portions of the program.

CSTF - the cost function indices in order of application

TOLER - the tolerance associated with each cost function

NF - the number of cost functions used

MAXSTAR - the maximum number of elements in a partial star.

ALTER - the number of new elements which are generated from one formula in a partial star P_i when forming a new partial star P_{i+1}.

EXTNTY - a flag indicating whether EXTNTY type predicates have been added.

EQUIV - a flag indicating whether EQUIV type predicates have been added

NCONSIST - the minimum number of consistent generalizations produced.

2.10 Arithmetic Expression Variables

Arithmetic expressions are parsed by VLINT using the second half of the parse table. VLINT is passed the starting row in the parse table where it is to start parsing. For arithmetic expressions, this row is a constant defined as ARITHDD. Arithmetic expressions are parsed onto an ARITHSTACK in reverse polish notation. The program uses a grammar which actually causes the order of execution to be from right to left. The ARITHSTACK entry contains the following fields:

ACTION - is a code telling what to do with this entry. It takes on the values ADD (perform addition), SUBTRACT (perform subtraction), MULTIPLY (perform multiplication), DIVIDE (perform division), MODULO (perform modular division), MINUS (perform a unary minus), FUNCT (this entry is a function to look up the value of), and NUMBER (this entry is an integer). If ACTION is an operator, then the other fields of the record are meaningless.

ARGUMENT - if ACTION is NUMBER then this field contains the integer value of the integer. If ACTION is FUNCT then this field contains the PNO (index into symbol table) of the corresponding function or predicate. During the computation process in CALCARITH, the ARGUMENT fields are updated to point to the graph index of the corresponding function or predicate in TOFIND.

DUMMY - is an array of pointers to the symbol table for each dummy variable of the function or predicate in ARGUMENT. It is only meaningful if ACTION is FUNCT. The function is assumed to have ordered dummy variables (ORDIRR is FALSE). The list is terminated by a zero index.

2.11 Additional Variables

INFILE - an integer specifying whether input is from the terminal or from CFILE.

NMQ - the number of elements in MQ

FREEG - pointer to the list of available graph structures

RESTLIST - pointer to the list of restrictions

STAR - pointer to the list of formulas in a star

MQ - pointer to the list of consistent formulas

GSET - pointer to the list of input formulas

CCVSET - pointer to the list of output formulas

STP,TRACE - sets of values for trace features

FIXIT - patch for compiler bug on DEC-10 PASCAL (fails to pass arguments which are sets by reference properly).

3. I/O Files

3.1 TABLES

This file contains the parse table information. Terminals in the grammar which are characters immediately follow any number (i.e. non-terminal). The end of each row of the parse table has a 0 followed by a (up to) 60 character name which describes this production (for use in printing error messages). The boolean array CONT has the value 1 if true, 0 if false. Below is the parse table as it currently stands:

```
CONT    SRULE    RHS      NAME
  <blank line>
0 1 3 -3 0A VL2 DECISION RULE
1 2 -3 0A VL2 DECISION RULE
0 3 -4 => -6 0A VL2 DECISION RULE
0 4 -6 -4 0A CONJUNCTION OF SELECTORS
1 5 -6 0A CONJUNCTION OF SELECTORS
0 18[ -19= -10 ] 0A VARIABLE SELECTOR
```

```
1 7[ -21( -14)= -10] 0A FUNCTION SELECTOR
1 10[ -21( -14)] 0A PREDICATE SELECTOR
1 7[ -21= -10] 0A NILADIC FUNCTION SELECTOR
0 8 -43, -10 0A LIST OF NUMBERS
1 9 -43.. -43 0AN INTERVAL OF NUMBERS

1 19* 0AN ASTERISK (SYMBOLIZING THE ENTIRE DOMAIN)
1 10 -43 0A SINGLE NUMBER
0 11 -19, -14 0A DEPENDENT VARIABLE LIST
1 20 -19. -14 0A DEPENDENT VARIABLE LIST (ORDER IRRELEVANT)
1 12 -19 0A SUBSCRIPTED VARIABLE
0 13 -19* -10; -17 0A LIST
1 14 -19= -10 0A LIST
0 15 2 0A SUBSCRIPTED VARIABLE
0 16 3 0A NUMBER
0 17 1 0A FUNCTION SYMBOL
0 32 -32= -23 0AN ARITHMETIC DERIVED DESCRIPTOR
0 25 -25 -37 -23 0AN ARITHMETIC EXPRESSION
1 31 -25 0AN ARITHMETIC EXPRESSION
0 25 -27 -39 -25 0A TERM
1 31 -27 0A TERM
0 31( -23) 0A FACTOR
1 31 -33 0A FACTOR
1 28- -33 0A FACTOR
1 31 -32 0A FACTOR
1 28- -32 0A FACTOR
0 31 -34( -35) 0A FUNCTION CALL
0 23 3 0A NUMBER
0 21 1 0A FUNCTION SYMBOL
0 31 -42, -35 0A LIST OF DUMMY VARIABLES
1 31 -42 0A LIST OF DUMMY VARIABLES

0 27+ 0AN ADDITION OPERATOR
1 29- 0AN ADDITION OPERATOR
0 24* 0A MULTIPLICATION OPERATOR
1 26/ 0A MULTIPLICATION OPERATOR
1 30% 0A MULTIPLICATION OPERATOR
```

```
0 22 2 0A DUMMY VARIABLE
0 5 -20 0A NUMBER
1 33 1 0A SYMBOLIC VALUE
```

3.2 EXPLAIN

This file contains text for explanation. Each explanation has a number and is delimited by a ! in column 1 followed by the number of the explanation preceeding the text and a ! in column 2 - 80 following the text. If a line ends with *, the program stops printing to allow the user to read the material. (See appendix A for a listing of this file).

3.3 CFILE

This file contains a set of input commands and data which is to be executed before the system asks for user input. Normally, input rules and certain parameters are included in this file.

3.4 VL1EVE

This file contains a list of VL type events. The file is in the format for AQ7 except that each event specification is preceeded with the class number of the associated decision. A -1 indicates a value which is irrelevant.

3.5 Other Files

IFILE and OUTPUT are the TTY input and output (these are TTY in the DEC 10 version). All other files are not currently used.

4. Program Structure

The program INDUCE_1 (Appendix C) contains about 4000 PASCAL statements and 40 basic procedures. These procedures may be grouped into several classes: 1) control and user interface, 2) VL to internal formula representation, 3) graph manipulation, 4) add new functions, 5) AQ7 complex manipulation and 6) supporting procedures. Each group of procedures operates nearly independently of the others thus giving the possibility of implementation on a smaller machine.

The main program accepts high level commands and calls the appropriate procedures to perform the requested action. Any input

in the form of a decision rule passes through the VLINT procedure for translation to internal format. On some occasions, information is then copied from one internal form to another (E command) but most of the work is done in VLINT. All other user interaction takes place in ENTERP (enter parameters). The VL_1 mode uses the VL_1 procedure and AQ, bypassing all procedures dealing with graph manipulation. To cover a set of formulas, the COVER procedure is called which in turn, calls NEWGP to grow generalizations and AQSET to apply AQ to the consistent generalizations in NQ.

4.1 Control and User Interface

MAIN - process high level commands

ENTERP - Decode commands using the first 4 characters of the command name. If it's a number, find a rule with that number in the rule base. Find the first two numbers in the command (GETNUM) and place in the variables I and L. Then, execute the command.

PGRAPH - Print the graph structure as VL formula. Assign indices to all variables. Write out function and arguments if any. Then, write out reference (if not *) If tree structured domain and the value is an internal node, then only print out the internal node.

PCPX - Print in VL type format indexing into SYMTAB using AQ.SLOC array to find the maximum and minimum values. Don't print any selector with a (*) reference.

PMETAD - Print list of selected meta-functions.

PDOM - Print domain table (i.e. dump symbol table).

EXPLN - Find requested text from the file EXPLAIN and print it stopping at (*) for carriage return from user.

4.2 VL Translation to Internal

TOKEN - Read an input line and add the terminator (?). Scan over the letters and digits and set CTYPE (0-delimiter, 1-function symbol, 2-variable, 3-number). If CTYPE was 0 then determine internal representation of the delimiter. If CTYPE is 1 or 2, then find the row in the symbol table (FINDROW). If it is not there, then add a new row to the symbol table (FIXSYM) (The name of the symbol is located between FCURS and LCURS in BUF). In the case of a variable, add an extra row for the domain of the variable in addition to a new row for the variable itself (i.e. a row for X in addition to a new row for X1). If CTYPE is 3, then compute the value of the number. Return the location in the symbol table or the computed number in the parameter SROW and delimiter type in CTYPE.

VLINT - Translate VL formula into graph structure. Maintain a value stack (VSTK), a function stack (FSTK), semantic stack (SSTK) and a parse stack (PSTK).

PSTK - Contains a stack of all non terminals not yet completed.

SSTK - Contains the tokens from the input buffer which have not been matched with an element of a completed production.

VSTK - the stack of numbers not already placed into the graph.

FSTK - the stack of arguments of a function (FSTK[1] is always the function symbol of the selector being parsed).

As tokens are accepted from the input buffer, they are matched with productions in PT. If a token does not match an element of a production which is a non terminal, the location of the non terminal is placed on PSTK and the production defining the nonterminal is tried (PROD and LOC determine the current element in PT under consideration). If there is no match, then try an alternative definition of the non terminal. If there is no alternative, back down PSTK and try another alternative of this non terminal.

If a token matches the element of PT under consideration, put this token on SSTK and try the next element in the production. If the complete production is matched, replace the matching tokens on SSTK with the appropriate nonterminal, back down PSTK to the previous location, process the indicated semantic rule (PROCESS) and proceed. Once the productions in row 1 of PT are completed, the expression is said to be syntactically correct.

PROCESS - Execute the semantic rule for the production (-PROD). Briefly, node assignments are made using the elements in PSTK, values in the reference are assigned from elements in VSTK. The MNVAL and EVAL fields of the symbol table are updated and the type of a node is determined. Links between variables and functions are assigned recalling that PSTK[1] contains the location of the function.

PARSEARITH - Execute semantic rules for arithmetic derived descriptors. A data structure called an ARITHSTACK is built which contains the arithmetic expression in reverse polish notation. The first element on the stack is the new variable to which the expression value should be assigned.

4.3 VL Formula Manipulation

SUBG1 - Determine if the graph in G1 is a subgraph of the graph in G2. If ALLSUBG is 1, then find all subgraphs of G2 which match G1 and apply ADDCONS (for restrictions). If ALLSUBG is 2, then find all subgraphs of G2 which match G1 and apply ALLC (AQ7 procedure). If ALLSUBG is 3, then find all subgraphs of G2 which match G1 and apply CALCARITH to compute the value of the arithmetic expression and add it to the graph. The procedure SUBG1 selects a starting node of G1 and a matching node of G2. SUBG produces a spanning tree of G1 from the starting node calling MATCH to determine for each pair of nodes whether they MATCH. For each pair of matching nodes, ASSIGN records the correspondence. If INSD is true, two nodes (selectors or

variables) are matched only if the values of the first cover the values of the second. If INSD is false, the values of the two nodes need only intersect.

TRIMG - Trim a list of formulas to MAXS elements, return other formulas to FREEG. Place formulas with COST[3] into MQ (consistent formulas). Instead of sorting a linked list, the array CA is sorted. Costs are assumed to be stored with each formula (calculated in COVER).

COSTG - Determine the cost function CT specified for the formula P.

COVER - Cover the set of formulas ES. First, select an element of F1 to cover (G) and compute the initial partial star. For all nodes in a graph, the flag COUNT is set to 1. Trim the partial star and apply absorption. Form a new partial star by calling NEWGP for each remaining element of the trimmed partial star. Once NCONSIST elements are in MQ, apply AQ7 (via AQSET) to each consistent formula. Trim the list to one best element and remove elements of F1 covered by this formula (set FP to false). Select a new element of F1 and repeat until F1 is exhausted.

NEWGP - Add new selectors to the input graph to form a list of ALTER or less new formulas. G0 is the old generalization of G1; direct association exists betwen nodes of G0 and nodes of G1 (i.e. correspondence is 1-1 by row, not through ASSGN as with other correspondences). The procedure forms only connected new graphs. A list of selectors which may be connected to the current graph is created in CANDID and sorted with respect to VCOST and NARG. All variables connected to existing nodes are flagged (COUNT=2) and then all function nodes connected to variables with COUNT = 1 or 2 are marked (COUNT=3) All count = 3 selectors are placed in CANDID. Then, a new graph (in SLST) is formed from the old one with a new selector and any relevant variables. EQUIV type functions are discarded if they have no more than 1 argument. The list SLST is returned to the calling procedure (COVER).

4.4 AQ7 Complex Manipulation

AQ - perform the AQ algorithm on the sets P1 and P2 of complexes obtained from the sets P1 and P0 of rules. This routine is much like AQ7 and is not further explained here.

LQST2 - perform the LQST function during characteristic generalization. During characteristic generalization, it becomes necessary to have a minimum sized cover which covers all rules (not complexes) in P1. Since there is often a many-to-one relationship between complexes in P1 and rules in P1, this is a non-trivial task and LQST2 performs this task. During the ALLC procedure, a CPXTABLE is attached to each complex which lists the rule numbers of the original rules in P1 which the complex covers. LQST2 loops finding the complex which covers the most rules, combining its reference values with the complex currently being derived and eliminating all complexes which cover the rules it covers from further consideration. When the set of complexes is exhausted, a quasi-minimal cover has been found. Trace 7 causes various information to be printed out during this covering process.

AQSET - Translate from VL representation (graph structure) to VL representation (sequence of sets of values). Create two sets of complexes, P1 containing subgraphs of graphs with VL set P1, and P2, the set of complexes associated with c-structures (GSUB) isomorphisms with elements of the VL set P0. The first element of P1 corresponds to the part of the graph GSUB which was consistent. The two sets of events are passed to the AQ procedure which returns a complex covering the first element of P1 but no element of P2. This is copied back into GSUB to form the extended reference generalization.

ALLC - Translate from graph to complex and add to the list of complexes if not already there. Also, set up SLOC to relate VL variables to symbols and find NVAR (number of

variables). Use assignments from the c-structure GSUB and the graph G1 for nodes with COUNT = 1 in GSUB. All meta-selectors are loaded in the first METATRIM VL variables, the remainder are nodes with COUNT = 1 in GSUB. A CPXTABLE is maintained for each complex which contains a list of the rule numbers of the rules which that complex covers. This is used by the LQST2 routine.

VL1 - Input VL events from the file VL1EVE and translate to complex storage. Call AQ to find generalization and then print result.

TRIMP - Trim a list of complexes with respect to AQCSTP etc. This is nearly the same as TRIMG but uses CPX structures.

COSTP - compute the cost of a complex.

4.5 Add New Functions

ADDSEL - find sets of nodes which have the same label in the graph. Add a new selector with the same label except that ORDIPR = true and PNO is the negative of the original PNO. The negative PNO always indicates a predicate of this type.

ADDML - Add MST, LST type EXTMTY predicates. For each binary predicate whose arguments assume values from the same domain, add extremity predicates.

ADDMETA - add meta-selectors to each formula in F1 and F0 For each unary function and function value, court the number of occurrences of this pair in a formula and add a selector of that type to the formula (COMPMS). Calculate F1COV and F0COV and sort the list of meta selectors (TRIMM).

PROCARITH - loop thru F1 and F0 adding an arithmetic derived descriptor to each graph in turn. This is accomplished by first creating a temporary graph (TOFIND) which contains the necessary functions and dummies from the right hand side of the arithmetic derived descriptor rule. This forms a connected graph structure. Then we call SUBG1 to find all isomorphisms between TOFIND and the rules in the

rule base. SUBG1 calls CALCARITH which actually performs the insertions into the rules. PROCARITH contains the internal procedure BUILDG which builds a graph corresponding to the arithmetic expression.

4.6 Supporting Routines

ILINE - input a new line from CFILE or the terminal

GETCHAR - read one character from the TTY or CFILE (perform ILINE if necessary)

PEOS - detect end of line on TTY or CFILE

INSIDE - determine if the set V_1 is a generalization of the set V_2 If INSD is TRUE the references of V_2 must completely cover those of V_1. If INSD is FALSE, the references need only intersect.

EXTND - find the extension of V_1 against V_2.

INIT - initialize variables and files

NEWG - allocate new graph.

GIN,GOUT,SOUT - not used

ADDCONS - add decision part of restriction (called from SUBG). Also used to perform exchange of one VL expression for another (see X command).

APPENDIX A

THE FILE EXPLAIN

EXPLAIN
11

THE PROGRAM HAS SELECTED AN EVENT E1 OF THE SET F1 WHICH HAS NOT BEEN COVERED YET. FIRST, A LIST OF C-FORMULAS EACH CONTAINING ONE SELECTOR WITH A UNARY FUNCTION WILL BE GENERATED. THIS LIST WILL BE TRIMMED TO VLMAXSTAR C-FORMULAS USING THE COST CRITERIA FOR THE VL PART OF THE PROGRAM. DURING TRIMMING, THE CONSISTENT FORMULAS ARE PLACED INTO THE MQ LIST (I.E. FORMULAS WITH COST FN 3 = 0). IF LESS THAN NCONSIST C-FORMULAS ARE IN THE MQ LIST, EACH ELEMENT OF THE PARTIAL STAR IS USED TO GENERATE A NEW LIST OF ALTERNATIVES EACH WITH ONE MORE SELECTOR THAN WAS IN THE PREVIOUS ELEMENT OF THE PARTIAL STAR. A SELECTOR IS ONLY ADDED TO A PRODUCT IF THE RESULT IS A CONNECTED GRAPH STRUCTURE. IF THE USER WISHES TO LIMIT THE NUMBER OF ALTERNATIVE PRODUCTS PRODUCED FROM ONE C-FORMULA, THIS LIMIT MAY BE SPECIFIED BY SUPPLYING A NON-ZERO VALUE TO THE PARAMETER ALTER.

ONCE AT LEAST NCONSIST CONSISTENT C-FORMULAS HAVE BEEN PRODUCED, THE AQ ALGORITHM IS APPLIED TO EACH FORMULA TO EXTEND THE REFERENCES OF SELECTORS AS MUCH AS POSSIBLE WHILE MAINTAINING CONSISTENCY. THEN THE BEST C-FORMULA IS SELECTED (LQ) AS THE COVER. SEE HELP TRACE UNDER THE'P' OPTION FOR AN EXPLANATION OF THE TRACE FUNCTIONS.*

UNTRIMMED PARTIAL STAR

THE FOLLOWING C-FORMULAS REPRESENT THE LIST OF ALTERNATIVE POSSIBLE CONSISTENT FORMULAS. ALONG WITH EACH FORMULA, THE COST FUNCTION VALUES FOR THE FORMULA ARE PRINTED IN THE ORDER OF EVALUATION. THESE FORMULAS WERE GENERATED BY ADDING A SELECTOR TO A PREVICUS INCONSISTENT FORMULA OR AT THE OUTSET, THIS IS A LIST OF SELECTORS OF E1 WITH UNARY FUNCTIONS. ALL OF THESE FORMULAS HAVE A CONNECTED GRAPH STRUCTURE REPRESENTATION. IN ADDITION, ANY EQUIVALENCE TYPE SELECTOR (I.E.[SH(X1,X2)=SAME]) IS REQUIRED TO HAVE AT LEAST TWO ARGUMENTS.

SELECTORS ARE ADDED TO A PRODUCT C1 USING THE FOLLOWING ALGORITHM@D

1 ALL VARIABLES (I.E. ARGUMENTS) WHICH ARE CONNECTED TO

SELECTORS IN THE PRODUCT C1 ARE LOCATED.

2 ALL SELECTORS WHICH ARE CONNECTED TO ANY VARIABLE IN 1 BUT NOT IN C1 ARE STORED IN A LIST. THIS LIST IS SORTED WITH RESPECT TO VCOST.

3 IF ALTER IS NOT 0, THEN THE LIST FROM 2 IS TRIMMED TO ALTER SELECTORS.*

4 FOR EACH SELECTOR IN 3, A NEW C-FORMULA IS CREATED WITH ALL SELECTORS IN C1 AND THIS SELECTOR. ALL RELEVANT LINKS BETWEEN SELECTORS AND VARIABLES ARE INCLUDED. IF AN EQUIVALENCE TYPE SELECTOR HAS ONLY ONE VARIABLE IN THE LIST FROM STEP 1, THE NEW GRAPH IS NOT ADDED TO THE NEW STAR LIST. OTHERWISE, A NEW STAR LIST IS FORMED WITH ALL THESE ALTERNATIVES.*

TRIMMED PARTIAL STAR

THE FORMULAS IN THE PARTIAL STAR ARE TRIMMED TO A SMALL LIST (MAXSTAR ELEMENTS) USING THE COST CRITERIA. THOSE FORMULAS WHICH ARE CONSISTENT ARE PLACED INTO THE MQ LIST. C-FORMULAS ARE SELECTED ACCORDING TO THE FOLLOWING PROCEDURE

1. FOR EACH COST CRITERION (IN THE ORDER SPECIFIED), EVALUATE THE COST OF ALL C-FORMULAS.

2. SELECT THE BEST MAXSTAR FORMULAS (I.E. THOSE WITH LOWEST COST) AND INCLUDE ALL FORMULAS WITH EQUIVALENT COST. TWO FORMULAS ARE EQUIVALENT IN COST IF THEY ARE WITHIN A TOLERANCE OF EACH OTHER. TOLERANCE MAY BE SPECIFIED IN ONE OF TWO WAYS FOR EACH COST CRITERION. AN INTEGER TOLERANCE IS AN ABSOLUTE VALUE, A TOLERANCE BETWEEN 0 AND 1 IS A RELATIVE TOLERANCE. AN ABSOLUTE TOLERANCE CAN BE GENERATED FROM A RELATIVE TOLERANCE BY COMPUTING THE MAXIMUM AND MINIMUM COST VALUES IN THE LIST OF FORMULAS (MAX AND MIN RESPECTIVELY) AND ASSIGNING THE ABSOLUTE TOLERANCE AT@L

$$AT = TOLERANCE*(MAX-MIN)$$

3. THE MAXSTAR BEST FORMULAS ALONG WITH EQUIVALENT FORMULAS ARE RETAINED AND THE REMAINDER OF THE FORMULAS ARE REMOVED FROM THE LIST.

4. THE LIST OF FORMULAS IS EVALUATED USING THE NEXT COST CRITERION. WITH THE LAST CRITERION, ONLY THE BEST MAXSTAR FORMULAS ARE RETAINED.!

!2

THERE ARE NOW AT LEAST NCONSIST ELEMENTS IN THE MQ LIST (OR THE PROGRAM CAN NOT GENERATE ANY MORE ALTERNATIVES). THE AQ PROCEDURE IS APPLIED TO THESE CONSISTENT FORMULAS. EACH FORMULA IS PRINTED BEFORE THE AQ PROCEDURE AND THEN THE RESULT AFTER AQ IS PRINTED. THE COST FUNCTION 1 IS BE EVALUATED FOR THESE FORMULAS.

!

!3

THE BEST FORMULA IN THE MQ LIST(LQ) IS SELECTED BY TRIMMING THE LIST OF FORMULAS WITH A MAXSTAR OF 1.

!

!4

THE AQ PROCEDURE IS APPLIED TO A SET OF VL1 EVENTS WHICH ARE DERIVED FROM A CONSISTENT C-FORMULA AND THE SET OF EVENTS IN F1 AND F0. BELOW, THE C-FORMULA STRUCTURE AND INPUT EVENTS ARE LISTED. THE VL1 VARIABLES CORRESPONDING TO THE NODES IN THE GRAPH OF THE C-FORMULA ARE GIVEN. IT IS KNOWN THAT THERE IS A CONSISTENT C-FORMULA WITH THE GIVEN STRUCTURE (I.E. THERE ARE VALUES FOR THE REFERENCES SO THAT THE FORMULA IS CONSISTENT). THE VL1 EVENTS REPRESENT DIFFERENT POSSIBLE SETS OF VALUES IN THE REFERENCE OF C-FORMULAS WITH THE SAME STRUCTURE IN EVENTS OF F1 AND F0. WE WANT TO INCLUDE AS MANY SUCH SETS OF VALUES WHICH CORRESPOND TO EVENTS IN F1 AND TO EXCLUDE ALL SUCH SETS WHICH CORRESPOND TO EVENTS OF F0. THE EVENTS OF SET 1 BELOW INCLUDE SETS ASSOCIATED WITH EVENTS IN F1. EVENTS OF SET 2 BELOW INCLUDE SETS OF REFERENCE VALUES ASSOCIATED WITH EVENTS IN F0.

!

!18

AT THIS POINT, YOU MAY CHANGE SOME PARAMETERS, SEE A RULE IN THE MEMORY, OR SEE THE CURRENT PARAMETERS. IN ORDER TO CHANGE A PARAMETER, ENTER THE PARAMETER NAME FOLLOWED BY THE PROPER SPECIFICATIONS. SOME PARAMETERS REQUIRE NO VALUES (PRULE), SOME REQUIRE ONE (TRACE) AND SOME

REQUIRE 2. IN GENERAL, ALL YOU HAVE TO DO IS ENTER THE
FIRST FOUR LETTERS OF THE PARAMETER NAME, THEN THE VALUE OR
TWO VALUES AS INTEGERS. ANY DELIMITERS MAY BE USED. ONE
EXCEPTION TO THIS IS THE PARAMETER VCOST WHICH MUST BE ENTERED
IN A PARTICULAR FORMAT. FOR FURTHER EXPLANATION OF THE
PARAMETERS AND WHAT THEY DO, TYPE

HELP <PARAMETER NAME>

TO SEE A RULE IN THE MEMORY, JUST ENTER THE RULE NUMBER.
TO RETURN TO WHAT YOU WERE DOING, ENTER

QUIT

!

!!CO

TRACE PARAMETER

THIS PARAMETER MAY HAVE A SET OF VALUES FROM 1 TO 10. EACH VALUE
RELATES TO A TRACE OF A PARTICULAR FEATURE OF THE PROGRAM. THE
VALUES CURRENTLY MEANINGFUL ARE THE FOLLOWING@D

1 PRINT ALL OF THE C-FORMULAS WHICH ARE GENERATED FROM
A PREVIOUS LIST OF C-FORMULAS. AT THE BEGINNING, ONLY
C-FORMULAS INVOLVING A SINGLE SELECTOR WITH A UNARY FUNCTION
ARE GENERATED. ON SUBSEQUENT PASSES THROUGH THIS TRACE, NEW
SELECTORS ARE ADDED TO THE THOSE FORMULAS REMAINING AFTER TRIMMING
WHICH FORM CONNECTED GRAPH STRUCTURES. IF ALTER IS NOT 0,
THEN ONLY AT MOST ALTER NEW FORMULAS ARE ADDED.
PRINT THE FORMULAS LEFT AFTER TRIMMING. DURING TRIMMING,
ALL CONSISTENT FORMULAS ARE REMOVED FROM THIS LIST AND PLACED
IN THE HQ LIST FOR SUBSEQUENT PROCESSING BY THE AQ ALGORITHM.
THESE MAY BE LISTED BY USING TRACE 2 BELOW.

2 PRINT ALL CONSISTENT FORMULAS. EACH FORMULA IN THE HQ
LIST IS PRINTED BEFORE AQ GENERALIZATION AND THEN THE
RESULTING FORMULA AFTER AQ GENERALIZATION IS PRINTED.

3 AFTER FULL GENERALIZATION, THE BEST HQ IS SELECTED (LQ)
AND PRINTED WITH THIS TRACE FEATURE. THE NEXT EVENT FROM
F1 IS THEN SELECTED AND THE ENTIRE PROCESS IS REPEATED.
THE FINAL COVER IS ALWAYS PRINTED.

4 ALL INPUT EVENTS TO THE AQ PROCEDURE ARE PRINTED WITH WITH THIS TRACE. ON THE FIRST PASS, THESE MAY NOT BE ALL THE EVENTS AND THEREFORE THE EVENTS ARE PRINTED FOR EACH PASS THROUGH THE AQ PROCEDURE.

5 THE SELECTED COMPLEX FROM THE CURRENT PASS THROUGH THE AQ PROCEDURE IS PRINTED IN AQ FORMAT.

6 PRINT THE SELECTED META FUNCTIONS

7 PRINT THE CHARACTERISTIC GENERALIZATION PROCESS DURING THE LQST2 PROCEDURE.

8 NOT USED

9 PRINT ALL ALTERNATIVE GENERALIZATIONS OF THE EVENT

10 PRINT EVENT E1 WHICH IS TO BE COVERED

TO TURN ON ANY TRACE FEATURE, ENTER

TRACE I

WHERE I IS THE NUMBER OF THE TRACE FEATURE WHICH IS TO BE TURNED ON.
TO TURN OFF THE TRACE FEATURE,ENTER

TRACE -I

WHERE I IS THE NUMBER OF THE FEATURE WHICH IS TO BE TURNED OFF.
TO STOP THE PROGRAM AT EACH TRACE FEATURE (POSSIBLY TO CHANGE SOME PARAMETERS), YOU MAY ENTER

STP I

WHERE I IS THE ASSOCIATED TRACE FEATURE. THE STOP MAY BE REMOVED BY ENTERING

STP -I

!
!200

AQCUTP1

IN ORDER TO SPEED UP THE AQ PROCEDURE, ONLY CUTP1 EVENTS ARE CONSIDERED IN THE COST FUNCTION 3. THE DEFAULT VALUE IS 20 BUT MAY BE CHANGED BY ENTERING

AQCUTF1 I

WHERE I IS THE NEW VALUE OF AQCUTF1

!

!3C0

AQMAXSTAR

THE AQMAXSTAR PARAMETER IS THE MAXSTAR PARAMETER USED IN THE AQ PROCEDURE. THIS SPECIFIES THE NUMBER OF ALTERNATIVE COMPLEXES IN THE CURRENT PARTIAL VL1 TYPE STAR.

!

!4C0

AQTOLERANCE

THIS PARAMETER SPECIFIES THE TOLERANCE FOR THE ITH COST FUNCTION. IF IT IS AN INTEGER, THEN IT IS ASSUMED TO BE AN ABSOLUTE VALUE; IF IT IS A VALUE BETWEEN 0 AND 1 THEN IT IS A RELATIVE VALUE WHICH IS CALCULATED BY DETERMINING THE MAXIMUM AND MINIMUM COST FUNCTIONS IN THE STAR AND THEN OBTAINING AN ABSOLUTE VALUE WHICH IS CALCULATED AS FOLLOWS@D

ABSOLUTE VALUE = TOLERANCE * (MAX - MIN)

ALL COMPLEXES WITHIN THE STAR WHICH HAVE COSTS WITHIN ABSOLUTE VALUE TOLERANCE ARE CONSIDERED TO BE EQUIVALENT WITH RESPECT TO TRIMMING.

THIS VALUE IS SPECIFIED BY ENTERING

AQTOLERANCE(I)=T

WHERE I MEANS THAT THIS TOLERANCE IS ASSOCIATED WITH THE ITH COST FUNCTION AND T IS THE TOLERANCE IN HUNDRETHS (IT MUST BE AN INTEGER) FOR EXAMPLE@D

AQTOLERANCE(2)=200

SPECIFIES THAT ALL COMPLEXES WITH THE SECOND COST FUNCTION VALUE WITHIN 2 ARE EQUIVALENT.

THE SYNTAX IS SOMEWHAT RELAXED TO REQUIRE ONLY THE FIRST FOUR LETTERS OF THE PARAMETER NAME (E.G. AQTO) AND THEN TWO NUMBERS WITH ANY DELIMITERS WHICH YOU DESIRE.

E.G. AQTO 2 200

IS INTERPRETED THE SAME AS THE ABOVE EXAMPLE.

!

!500

AQCRIT

THIS PARAMETER SPECIFIES THE ORDER OF APPLICATION OF COST CRITERIA. FOR THE AQ PROCEDURE. SIX CRITERIA ARE CURRENTLY AVAILABLE@D

1 THE NUMBER OF NEW VL1 EVENTS WHICH ARE COVERED. ALTHOUGH THIS IS NOT THE NUMBER OF C-FORMULAS WHICH ARE COVERED, IS MAY BE A CLOSE APPROXIMATION IN CERTAIN CASES AND RUNS MUCH MORE QUICKLY THAN COST 3

2 THE NUMBER OF SELECTORS IN A COMPLEX WHICH DO NOT HAVE * IN THE REFERENCE

3 THE NUMBER OF C-FORMULAS WHICH ARE ACTUALLY COVERED BY THIS COMPLEX. THIS IS MORE TIME CONSUMING THAN 1 BUT MAY GIVE BETTER RESULTS DEPENDING ON THE PROBLEM.

4 THE SUM OF THE COSTS OF VARIABLES IN THE COMPLEX.

5 THE NUMBER OF EVENTS COVERED IN THE VI1 SET 1

6 THE NUMBER OF EVENTS COVERED IN THE VI1 SET 2

THIS PARAMETER MAY BE ENTERED BY TYPING

AQCRIT(I) = J OR AQCRIT(I) = -J

WHERE I SPECIFIES THE ORDER OF EVALUATION OF THIS CRITERION AND J IS THE CRITERION (I AND J IN THE INTERVAL [1..6]. THE FORMAT OF THIS SPECIFICATION MAY BE RELAXED TO ONLY SPECIFY THE FIRST FOUR LETTERS OF THE PARAMETER NAME (AQCR) AND THEN TWO NUMBERS, I AND J.

!

!600

AQNF

THIS PARAMETER SPECIFIES THE NUMBER OF AQ COST CRITERIA WHICH ARE TO BE USED. IT MUST BE IN THE INTERVAL [1..6]

!

!700

VCOST

THIS PARAMETER SPECIFIES THE COST OF A VARIABLE. INITIALLY, ALL VARIABLES HAVE COST OF 0. TO CHANGE THE COST OF A VARIABLE, ENTER

VCOST(<VARIABLE NAME>)=II

WHERE VARIABLE NAME IS THE NAME OF THE VARIABLE (OR DESCRIPTOR) WHICH IS USED IN THE RULES. II IS THE COST OF THIS VARIABLE (IT MAY BE NEGATIVE). THE SYNTAX IS IMPORTANT HERE, YOU MUST USE LEFT AND RIGHT BRACKETS '(..)' AND LEAVE NO SPACES.

EXAMPLE@D VCOST(SHAPE)=-2

SETS THE COST OF THE DESCRIPTOR SHAPE TO -2.

!

!8C0

VLMAXSTAR

THIS PARAMETER GIVES THE MAXSTAR PARAMETER FOR THE VL2 PART OF THE PROCEDURE. IT SPECIFIES THE NUMBER OF ALTERNATIVE C-FORMULAS WHICH ARE RETAINED IN A PARTIAL STAR IN EACH STEP.

!

!9C0

VLTOLERANCE

THIS PARAMETER GIVES THE TOLERANCE FOR THE ITH COST FUNCTION FOR C-FORMULAS IN THE VL2 TRIMMING PROCEDURE. IF IT IS AN INTEGER, THEN IT IS ASSUMED TO BE AN ABSOLUTE TOLERANCE, OTHERWISE IT IS RELATIVE TO THE MAXIMUM AND MINIMUM COSTS IN THE PARTIAL STAR. THE VALUE IS ENTERED IN HUNDREDTHS (SEE AQTOLERANCE).

EXAMPLE@D VLTOL(3)=200

SPECIFIES THAT THE THIRD VL2 COST CRITERION (VICRIT(2)) HAS AN ABSOLUTE TOLERANCE OF 2 (=2.00)

!

!1C00

VICRIT

THIS PARAMETER SPECIFIES THE ORDER IN WHICH COST CRITERIA ARE TO BE APPLIED IN TRIMMING OF C-FORMULAS. FIVE CRITERIA ARE CURRENTLY AVAILABLE@D

1 THE NUMBER OF EVENTS OF F1 COVERED

BY THIS C-FORMULA BUT NOT BY ANY PREVICUS LQ

2 THE NUMBER OF SELECTORS IN THE C-FORMULA.

3 THE NUMBER OF EVENTS IN FO CCVERED BY THE C-FORMULA

4 THE TOTAL SUM COST OF DESCRIPTORS IN SELECTORS. IF A DESCRIPTOR APPEARS MCRE THAN ONCE IN THE FORMULA, THEN IT IS COUNTED FOR EACH APPEARANCE, NCT JUST CNCE.

5 THE TOTAL SUM COST OF DUMMY VARIABLES IN SELECTORS. IF A DUMMY VARIAELE APPEARS MORE THAN CNCE IN THE FORMULA, THEN IT IS CCUNTED FOR EACH APPEARANCE. COST REFER TO THE ORIGINAL DUMMY VARIABLE WHICH WAS ENTERED BY THE USER (NCT THE SUBSCRIPT ASSIGNED BY THE PRCGRAM).

THIS PARAMETER IS SPECIFIED EY ENTERING
VLCRIT(I)=J
WHICH SPECIFIES THAT THE ITH CRITERICN IS NUMBER J ABOVE.
EXAMPLE@D VLCRIT(1)=3

!

!1100

VINF

THIS PARAMZTER SPECIFIES THE NUMBER OF COST CRITERIA WHICH ARE TO BE USED IN THE VL2 TRIMMING ANC SELECTICN PROCESS.

!

!1200

NCCNSIST

THIS SPECIFIES THE MINIMUM NUMBER CF CONSISTENT FORMULAS WHICH ARE TO BE GENERATED IN THE VL2 PART OF THE ALGCRITHM. EACH OF THESE C-FCRMULAS IS GENERALIZED BY THE AQ ALGORITHM.

DURING CHARACTERISTIC GENERALIZATICN, A C-FORMULA IS CONSIDERED TO BE "CCNSISTENT" IF

1 IT COVERS AT LEAST MINCOVER PERCENT OF THE FCRMULAS IN F1

2 ALL C-FORMULAS DERIVED FRCM IT (DURING THE GROWTH PROCESS) DO NOT CCVER AT LEAST MINCOVER PERCENT OF THE FCRMULAS IN F1.

!

!1300

ALTER

THIS PARAMETER REFERS TO THE GENERATION OF CONSISTENT FORMULAS AND SPECIFIES THE NUMBER OF NEW FORMULAS WHICH WILL BE FORMED BY ADDING SELECTORS TO AN EXISTING MEMBER OF THE PARTIAL STAR. ONLY NEW SELECTORS ARE ADDED WHICH WILL FORM A CONNECTED GRAPH STRUCTURE. EQUIVALENT SELECTORS ([SH(X1.X2)=SAME]) ARE ADDED ONLY IF THERE WERE TWO DUMMY OR INDEPENDENT VARIABLES IN THE ARGUMENT LIST OF THE SELECTOR IN THE ORIGINAL FORMULA OF THE PARTIAL STAR.

IF ALTER IS 0, THEN A NEW C-FORMULA IS GENERATED FOR ALL SELECTORS NOT YET USED IN THE CURRENT C-FORMULA AND WHICH FORM A CONNECTED SUBGRAPH.

!

!1400

PRINT

THIS PARAMETER REQUESTS A LIST OF THE META SELECTORS CURRENTLY SELECTED, THE DOMAIN STRUCTURES, THE INPUT RULES OR RESTRICTIONS. ENTER@D

PRINT M FOR META SELECTORS
PRINT D FOR DOMAINS
PRINT R FOR RESTRICTIONS
PRINT F FOR INPUT DECISION RULES.

!

!1500

METATRIM

THIS PARAMETER SPECIFIES THE NUMBER OF META FUNCTIONS SELECTED. IT SHOULD BE LESS THAN GSIZE. IF IT IS 0, THEN NO META FUNCTIONS ARE COMPUTED.

!

!1600

DESCTYPE

DESCTYPE INDICATES WHAT TYPE OF DESCRIPTION THE PROGRAM SHOULD GENERATE WHEN THE "C" (COVER) COMMAND IS ISSUED. THE PROGRAM CAN GENERATE TWO TYPES OF DESCRIPTIONS@D

1 DESCTYPE CHARACTERISTIC. A CHARACTERISTIC DESCRIPTION OF A SET OF RULES P1 IS THE MOST SPECIFIC DESCRIPTION WHICH RULES P1 IS SHARED BY

ALL EVENTS IN F1. FO MOSI BE EMPTY FOR THIS TC WORK PROPERLY. THUS, CNLY ONE SET CF EVENTS SHOULD BE SUPPLIED TO THE PROGRAM FOR CHARACTERISTIC DESCRIPTICN. THE PARAMETER, MINCOVER, MUST ALSO BE SET (SEE HELP MINCOVER).

2 DESCTYPE DISCRIMINANT. A DISCRIMINANT IESCRIPTION CF A SET CF RULES, F1 AGAINST ANCTHER SET OF RULES FO IS THE MCST GENERAL DESCRIPTION WHICH DISCRIMINATES RULES IN F1 FROM RULES IN FO. IT MUST NOT COVER ANY RULES IN FO AND IT IS DESIREABLE TEAT IT COVER AS MANY RULES IN F1 AS POSSIBLE (THIS IS THE IEFAULT).

NCTE@D APPRCPRIATE COST FUNCTIONS SHOULD ALSO BE SET FOR CHARACTERISTIC (-1, -2) AND DISCRIMINANT (3, -1, 2) IESCRIPTORS.

THIS COMMAND MAY BE ABBREVIATED TC

DESC C (FOR CHARACTERISTIC DESCRIPTICNS)
DESC D (FCR DISCRIMINANT IESCRIPTIONS)

!

!1700

QUIT

RETURN TO THE CCMMAND LEVEL. THE PROGRAM WILL RESUME FROM THE LAST POINT.

!

!1800

HELP

HELP GIVES A LIST OF ALL PARAMETERS WHICH ARE UNDERSTOOD AT THIS POINT

!

!1900

PARAMETERS

LIST CURRENT VALUES OF PARAMETERS

!

!2000

STP

HALT THE PRCGRAM AT A PARTICULAR TRACE FEATURE. GENERALLY, THIS MAY BE USED TO GET AN EXPLANATION OF WHATS HAPPENING OR TO CHANGE SCME PARAMETER.

!

!2100

MINCOVER

THIS SPECIFIES THE MINIMUM PERCENTAGE OF RULES IN F1 THAT A C-FORMULA MUST COVER IN ORDER TO BE CONSIDERED AS A CHARACTERISTIC DESCRIPTION. IT IS USED IN CONJUNCTION WITH NCONSIST TO DETERMINE WHEN THE C-FORMULA GROWING PROCESS SHOULD STOP. DURING THE C-FORMULA GROWING PROCESS, EACH C-FORMULA IS GROWN (BY ADDING NEW SELECTORS) UNTIL ALL OF THE FORMULAS WHICH CAN BE GROWN FROM IT FAIL TO COVER MINCOVER PERCENT OF THE RULES IN F1. AT THAT TIME, IT IS PLACED ON THE MQ. NCONSIST SUCH MQ RULES MUST BE FOUND BEFORE THE GROWING ALGORITHM TERMINATES. THUS, IF MINCOVER=100, SEVERAL FAIRLY TRIVIAL RULES WILL BE FOUND. IF MINCOVER=50, SOME INTERESTING RULES INVOLVING MANY SELECTORS WILL BE FOUND BUT THESE RULES MAY NOT COVER ALL OF F1.

!

!2200

QUICK

THIS TURNS OFF ALL TRACES

!

!2300

DETAIL

THIS TURNS ON ALL TRACES.

!

!2400

EXPLAIN

THIS TURNS ON ALL TRACES AND SETS ALL STOPS

!

!2500

BRIEF

THIS SETS TRACE OPTIONS 10 AND STOP OPTIONS 10

!

!2600

VTYPE

ENTER VTYPE IN THE SAME FORMAT AS VCOST. THE TYPES ARE:

1 - NOMINAL

2 - INTERVAL

3 - STRUCTURED

!

!2700

PRULE

THIS PARAMETER PRINTS THE RULES AS WELL AS THE RULE NUMBERS AT EACH STEP. TO SUPRESS PRINTING RULES, ENTER PRULE F. TO RESUME PRINTING RULES, ENTER PRULE. THIS MAY BE USED IF THE RULES ARE VERY LARGE AND REQUIRE A LONG TIME TO PRINT ON THE TERMINAL.

!

!2800

LOST

THIS PARAMETER (ON BY DEFAULT) STRIPS EACH OUTPUT COMPLEX FROM THE AQ7 PROCEDURE. TO TURN OFF, ENTER LOST F.

!

!5

THE RESULT OF THE AQ APPLICATION IS GIVEN BELOW. IF THIS IS NOT CONSISTENT, MORE EVENTS WILL BE ADDED TO SET 2 AND AQ REPEATED. IF IT IS CONSISTENT, THEN IT WILL BE TRANSLATED BACK INTO A VL2 FORMULA AND STORED IN THE NEW MQ LIST.

!

!6

THE SELECTED META FUNCTIONS ARE LISTED BELOW. HERE IS AN EXPLANATION OF THE TABLE.

MS IS AN INTERNAL NUMBER USED TO REFER TO THIS META FUNCTION.

TYPE IS #PT OR FORALL. #PT INDICATES THAT THIS FUNCTION MEASURES THE NUMBER OF OBJECTS FOR WHICH THE ASSOCIATED DESCRIPTOR TAKES ON THE ASSOCIATED VALUE. FORALL INDICATES THAT ALL OBJECTS IN A RULE FOR WHICH THE ASSOCIATED DESCRIPTOR IS APPLICABLE HAVE THE ASSOCIATED VALUE.

FUNCTION INDICATES THE DESCRIPTOR AND VALUE PAIR REFERRED TO ABOVE.

F1COV LISTS THE MAXIMUM NUMBER OF RULES IN F1 WHICH THIS META FUNCTION COVERS FOR ANY ONE SINGLE VALUE OF ITS REFERENCE.

FOCOV LISTS THE CORRESPONDING NUMBER OF RULES IN FO WHICH THIS META FUNCTION COVERS WHEN IT TAKES ON THE VALUE WHICH GIVES THE ASSOCIATED F1COV VALUE.

EXAMPLE@D

TYPE	FUNCTION	F1COV	FOCCV
#PT	SIZE = 2	5	2

THIS INDICATES THAT IN EACH RULE IN F1 AND FO THE PROGRAM IS CCUNTING THE NUMBER OF SELECTIONS OF THE FORM [SIZE(X)=2] (WHERE X IS ANY RELEVANT DUMMY VARIABLE). A NEW DESCRIPTOR, [# XS SIZE 2=K] WILL BE ADDED TO EACH EVENT. K IS THE NUMBER OF DUMMY VARIABLES FOR WHICH [SIZE(X)=2] IN THAT RULE. NOTICE K COULD BE 0 AS WELL AS A FINITE NUMBER. THE VALUE OF K WHICH COVERS THE MOST RULES IN F1 IS NOT LISTED IN THE TABLE, BUT IT COVERS 5 RULES IN F1 AND 2 RULES IN FO.

!

!7

THE PROGRAM IS NOW READY TO CHARACTERIZE A SET CF COMPLEXES. EACH COMPLEX HAS WITH IT A LIST OF THE RULES WHICH IT COVERS. THE PROGRAM DETERMINES THE COMPLEX WHICH COVERS THE MOST RULES AND REMOVES THOSE RULES FROM THE LISTS OF RULES COVERED BY THE REMAINING COMPLEXES. THIS PROCESS IS REPEATED UNTIL ALL RULES ARE COVERED. THE REFERENCES OF THE SELECTED COMPLEXES ARE MERGED TC GIVE THE FINAL COVER.

EXPLANATION OF THE OUTPUT@D

MAXCOUNT IS THE MAXIMUM NUMBER OF RULES COVERED BY ANY ONE COMPLEX.

WHEN MORE THAN ONE COMPLEX CCVERS MAXCCUNT RULES, THIS TIE IS RESOLVED BY COMPUTING THE COST OF ADDING EACH COMPLEX TO THE EMERGING COVER. THIS COST IS EQUAL TO THE NUMBER OF NEW REFERENCE VALUES WHICH WILL NEED TO BE ADDED IN ORDER TO MERGE THE COMPLEX AND THE COVER. THE COMPLEX WITH THE LOWEST COST IS SELECTED. THE PROGRAM PRINTS OUT THE EMERGING COVER AND THE SELECTED COMPLEX. IT ALSO PRINTS OUT THE CCST CF ADDING THE SELECTED COMPLEX TC THE EMERGING COVER.

!

!9

THESE ARE THE C-FORMULAS WHICH HAVE BEEN GENERALIZED BY THE AQ PROCEDURE. EACH ONE IS CONSISTENT (CCVERS NC RULES IN FO). ONLY THE BEST CONSISTENT C-FORMULA (MQ) WILL BE RETAINED, BUT ALL OF THESE FORMULAS MAY BE OF INTEREST TO THE USER.

!

!1C

AN EVENT E1 OF F1 HAS BEEN SELECTED. (F1 IS THE SET OF ALL CONDITIONS WHICH HAVE THE DESIRED SET IN THE DECISION PART; THE SET F0 IS THE SET OF ALL OTHER CONDITION PARTS KNOWN TO THE PROGRAM). THIS EVENT E1 WILL BE COVERED BY A C-FORMULA (CONNECTED CONJUNCTIVE VL2 FORMULA) WHICH IS CONSISTENT WITH RESPECT TO ALL FORMULAS OF F0 (I.E. COVERS NO FORMULA OF F0). ONCE A COVER (LQ) OF E1 IS FOUND, ALL EVENTS COVERED BY THIS LQ ARE REMOVED FROM F1 AND THE NEXT ELEMENT OF F1 IS SELECTED UNTIL NO MORE ELEMENTS CAN BE FOUND IN F1.

!

!21

ENTER RESTRICTIONS

THIS COMMAND ALLOWS THE USER TO ENTER RESTRICTIONS WHICH WILL BE APPLIED TO ALL THE EVENTS WHICH WILL BE INPUT LATER RESTRICTIONS SIMPLY ADD NEW INFOMATION TO THE EVENT BY APPENDING CERTAIN SELECTORS TO THE EVENT. THE INPUT FORMAT REQUIRES A PRODUCT OF SELECTORS WHICH FORM A CONNECTED GRAPH REPRESENTATION FOLLOWED BY '=>' AND A SELECTOR WITH A FUNCTION SYMBOL AND ARGUMENTS WHERE EACH ARGUMENT APPEARS IN THE CONDITION PART OF THE RULE SOMEWHERE.

EXAMPLE @D

[LEFT(X1,X2)][LEFT(X2,X3)]=>[LEFT(X1,X3)].

[STA(X1)=1][PART(X1,L1)]=>[COND(L1)=*].

!

!22

MODIFY RULES (EVENTS)

THIS COMMAND ALLOWS A USER TO ADD OR DELETE AN EVENT FROM THE SYSTEM. AFTER THE USER ENTERS THE CHARACTER M, THE PROGRAM ASKSS IF YOU WANT TO ADD OR DELETE A RULE. ENTER A OR D.

ADD A RULE

ENTER A, THEN ENTER THE RULE. THER RULE MAY BE BROKEN ACRSS SELECTOR BOUNDARIES IF IT WON'T FIT ON ONE LINE. IF YOU MAKE A MISTAKE, YOU MUST REENTER THE ENTIRE RULE FROM THE BEGINNING. SEE RULE SYNTAX BELOW.

DELETE A RULE

ENTER D. THE PROGRAM LISTS EACH EVENT KNOWN TO THE SYSTEM. AFTER EACH EVENT IS LISTED THE PROGRAM ASKS IF IT IS TO BE DELETED. ANSWER@D

Y - TO DELETE THE RULE

N - TO RETAIN THE RULE AND LIST THE NEXT ONE

Q - TO RETURN TO THE COMMAND MODE.*

RULE SYNTAX

A RULE CONTAINS A CONDITION PART (PRODUCT OF SELECTORS) AND A DECISION PART (A SINGLE SELECTOR WITH A 0-ARY FUNCTION OR DECISION VARIABLE) FOLLOWED BY A PERIOD (.). EACH SELECTOR IN THE CONDITION PART HAS A FUNCTION SYMBOL FOLLOWED BY ALIST OF ARGUMENTSS SEPARATED WITH ','. THE FUNCTION SYMBOL IS A NAME WITH LESS THAT 10 CHARACTERS. THE ARGUMENTS CONTAIN A NAME (THE NAME OF A GROUP OF COMPARABLE DUMMY VARIABLES) AND A NUMBER WHICH DISTINGUISHES THIS ARGUMENT FROM OTHERS OF THE SAME GROUP (E.G. X1 OR CAR4). THE REFERENCE MAY BE OMITTED (IN WHICH CASE IT ASSUMES THE VALUE 1), IT MAY BE * (ALL VALUES), A LIST OF INTEGERS SEPARATED BY COMMAS, OR A PAIR OF INTEGERS SEPARATED BY .. (THIS SPECIFIES A RANGE OF VALUES AND TELLS THE SYSTEM THAT THE FUNCTION HA'S AN INTERVAL DOMAIN STRUCTURE).

```
 SELECTOR EXAMPLES@D  [SH(X1)=1,2]  [P(X1,X2)]  [SH(A1)=*] [SIZE(L1)=1..6]
RULE EXAMPLE@D   [SH(X1)=3][Q(X1,X2)]=>[C=1,2].
```

!

!23

COVER A SET OF FORMULAS

THE SYSTEM WILL ASK WHICH SET. ENTER THE NUMBER WHICH IS THE DECISION VALUE WHICH IS TO BE GENERALIZED. YOU WILL PROBABLY WISH TO ENTER 'P' AND SET SOME TRACE AND STOP OPTIONS BEFORE ACTUALLY INITIATING THE COVER PROCEDURE. (SEE PARAMETERS QUICK,DETAIL,BRIEF ETC.)

!

!24

CHANGE PARAMETERS

ENTER P TO CHANGE PARAMETERS. ONCE YOU ARE IN THE PARAMETER MODIFICATION SECTION, TYPE HELP FOR FURTHER EXPLANATION. ALSO, WHEN THE PROGRAM STOPS DURING A TRACE, YOU MAY ENTER P TO GET THIS PROCEDURE.

!

!25

ENTER DOMAIN STRUCTURES

ENTER E AND THEN ENTER A RULE WITH FUNCTION SYMBOLS WITHOUT ARGUMENTS. ENTER THE LOWEST LEVELS OF GENERALIZATIN FIRST. ENTER E AND THEN

THE RULE FOR EACH GENERALIZTICN RULE.

EXAMPLE@D [SH=1,2,4]=>[SH=7].

!

!26

HELP

YOU MAY ENTER 'HELP X' WHERE X IS M,C,V,R,P,L,S, OR E IN ORDER TO OBTAIN AN EXPLANATIN OF EACH OF THESE COMMANDS.

!

!27

VL1 MCDE

ENTER THE VL1 MODE OF PROGRAM OPERATION WHICH BYPASSES VL2 CONSISTENT C-FORMULA GENERATION. YOU WILL BE ABLE TO ENTER VL1 EVENTS IN A MODIFIED AQ7 FORMAT FRCM A FILE VL1EVE. THE FORMAT OF THIS FILE CONTAINS A LIST OF EVENTS (VALUES OF VARIABLES) PRECEDED BY THE DECISICN VALUE. FOR EXAMPLE, IF THERE ARE TWO EVENTS IN SET 1 AND 2 EVENTS IN SET 5, THEN ENTER INTO THE FILE@D

1 0 1 3
5 1 1 3
5 1 1 2
1 1 1 1

IN THIS EXAMPLE THERE ARE THREE VARIABLES. NOTICE THAT THE ORDER OF EVENTS IS IRRELEVANT SINCE THE DECISION VALUE IS INCLUDED IN THE EVENT SPECIFICATION. THIS FILE MUST BE CREATED BEFORE RUNNING THE PROGRAM.

IN ORDER TO RUN THE PROGRAM IN VL1 MODE, CREATE A FILE IN THE ABOVE FORMAT CALLED VL1EVE. THEN RUN THE PRCGRAM AND ENTER V. AT THIS POINT, YOU MAY
ENTER DOMAIN STRUCTURES (IN THE VL2 FORMAT), ENTER PARAMETERS (THIS ALLOWS ONE TO ENTER COST FUNTICNS AND MAXSTAR PARAMETERS ETC.) OR COVER ONE SET AGAINST A BUNCH OF SETS OF EVENTS. *

VARIABLE COSTS AND DOMAIN TYPES (CHANGE DOMAIN TYPE FROM THE DEFAULT (NOMINAL) TO INTERVAL) MAY THEN BE ENTERED BY ENTERING P AND THEN SPECIFYING EITHER VTYPE OR VCOST PARAMETERS. ALL VARIABLES ARE LABELLED 'XI'. STRUCTURED DOMAINS ARE AUTOMATICALLY SET BY THE E COMMAND. THE DOMAIN TYPES ARE@D

1 - NOMINAL
2 - INTERVAL
3 - STRUCTURED

ONCE THE EVENTS ARE READ INTO THE PROGRAM AND ALL PARAMETERS ARE SET, YOU ARE READY TO COVER A SET OF EVENTS. ENTER THE C COMMAND. THE PROGRAM ASKS WHICH SET IS TO BE COVERED. ENTER THE NUMBER WHICH CORRESPONDS TO THE SET WHICH IS TO BE COVERED. THE PROGRAM THEN ASKS WHICH SETS ARE TO BE COVERED AGAINST. ENTER A LIST OF INTEGERS WHICH CORRESPOND TO THE SETS AGAINST WHICH THE COVER IS TO BE MADE. THE PROGRAM THEN PRINTS THE COVERING COMPLEXES.

ALL COMANDS EXCEPT FOR THE NUMBER OF VARIABLES AND SETS INVOLVED IN COVERING MAY BE ENTERED IN CFILE.

!

!28

L - EXTMTY PREDICATES

ADD EXTMTY TYPE PREDICATES LIKE LST- AND MST-

!

!29

S - EQUIV PREDICATES

ADD EQUIVALENCE TYPE PREDICATES (E.G [SH(X1,X2)=SAME])

!

!30

A - ENTER THE DEFINITION FOR AN ARITHMETIC DERIVED DESCRIPTOR

ENTER THE DERIVATION RULE FOR AN ARITHMETIC DERIVED DESCRIPTOR IN THE FORMAT@D

<NEW DESCRIPTOR>(<DUMMY VARIABLES>)= <ARITHMETIC EXPRESSION>.

EXAMPLE@D

GIRTH(X1)=LENGTH(X1)+WIDTH(X1).

THE DUMMY VARIABLE OF THE <NEW DESCRIPTOR> MUST APPEAR IN THE ARITHMETIC EXPRESSION. THE EXPRESSION IS WRITTEN IN STANDARD ALGEBRAIC FORM. THE OPERATORS WHICH MAY BE USED ARE@D

- \+ ADDITION
- \- SUBTRACTION OR UNARY MINUS
- * MULTIPLICATION

/ INTEGER DIVISION (REMAINDER DISCARDED)

% MODULO DIVISION

INTEGER CONSTANTS AND FUNCTIONS MAY ALSO APPEAR IN THE EXPRESSION. THE EXPRESSION MUST CONTAIN AT LEAST ONE FUNCTION OR PREDICATE AND MUST FORM A CONNECTED GRAPH STRUCTURE. ALL FUNCTIONS ARE PREDICATES ARE ASSUMED TO HAVE SINGLE VALUES IN THE REFERENCES. IF MORE THAN ONE VALUE APPEARS IN THE REFERENCE OF A FUNCTION WHEN THE EXPRESSION IS TO BE EVALUATED, THE SMALLEST VALUE IS USED. IF THE RESULTING COMPUTED VALUE LIES OUTSIDE THE RANGE OF VALID VALUES (0..MNVAL), THE DESCRIPTOR IS IGNORED AND NOT ADDED TO THE RULE. TO MAKE THE RIGHT HAND SIDE INTO A CONNECTED GRAPH, CONNECTING PREDICATES MAY BE MULTIPLIED TO THE EXPRESSION. ALL PREDICATES HAVE VALUE 1 WHEN THEY ARE TRUE.

EXAMPLE

SUMSIZE(X1,X2)=(SIZE(X1)+SIZE(X2))*P(X1,X2).

WHERE P IS SOME CONNECTING PREDICATE OF X1 AND X2.

TO PROCESS THE ARITHMETIC DESCRIPTOR DEFINITIONS ONCE THEY HAVE BEEN GIVEN TO THE PROGRAM, ISSUE THE "N" COMMAND. (SEE H N).

NOTE@D A GIVEN DESCRIPTOR PLUS DUMMY VARIABLE MAY ONLY APPEAR ONCE IN THE EXPRESSION ON THE RIGHT HAND SIDE OF THE RULE. THUS THE RULE@D

SQUARE(X1)=SIZE(X1)*SIZE(X1).

WILL NOT WORK.

!

!31

N - EXECUTE PREVIOUSLY ENTERED A COMMANDS.

THE N COMMAND CAUSES ALL PREVIOUSLY ENTERED ARITHMETIC DERIVED DESCRIPTOR DEFINITIONS (SINCE THE LAST N COMMAND) TO BE PROCESSED AND ADDED TO ALL RULES IN THE RULE BASE WHERE APPROPRIATE. THIS COMMAND IS PROVIDED SO THAT THE USER CAN ENTER THE ARITHMETIC DESCRIPTORS AT ANYTIME DURING THE PROCESS AND THEN APPLY THEM TO THE RULE BASE WHEN ALL OF THE RULES HAVE BEEN ENTERED.

!

!32

X - ENTER A LOGICAL DERIVED DESCRIPTOR AND SUBSTITUTE IT.

THE "X" COMMAND PERMITS THE USER TO ENTER A LOGIAL DERIVED DESCRIPTOR WHICH IS TO BE SUBSTITUTED (EXCHANGED) FOR ITS PREMISE IN EACH RULE IN WHICH THE PREMISE IS TRUE (THE PREMISE IS THE LEFT HAND SIDE OF THE DERIVATION RULE).

EXAMPLE@D

X

[BIG(PART1)][BOX(PART1)]=>[BIGBOX(PART1)].

THIS EXAMPLE WILL SUBSTITUTE THE PREDICATE [BIGBOX(PART1)] FOR EACH CONJUNCTION OF BIG(PART1) AND BOX(PART1) IN THE RULE BASE. PART1 REFERS TO ANY DUMMY VARIABLE IN THE "PART" FAMILY (WITH DIFFERENT SUBSCRIPT).

!