UIUCDCS-R-77-876

UILU-ENG 77 1727

INDUCE-1: An Interactive Inductive Inference Program in VL_{21} Logic System

by

James B. Larson

May 1977

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

INDUCE-1: An Interactive Inductive Inference Program in VL_{21} Logic System

by

James B. Larson

Department of Computer Science
University of Illinois
Urbana, Illinois

May 1977

This work was supported in part by the National Science Foundation, Washington, DC, grant no. NSF MCS 74-03514.

TABLE OF CONTENTS

CHAPTER PAGE

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 refered 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 preform 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_2 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:

```
M

A
```

```
[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 (carraige 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

```
R

[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_1 applications, this should be done after a V command has been entered since the V command initializes the symbol table for the special VL_1 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].
```

E,S (Enter ENTITY and EQUIV 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.)

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.) The number of variables and the classes to be covered and covered against must be entered from the terminal. All other 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 follwed 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_1 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.

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 MQ 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_2 c-structure and the VL_1 variables

5 - Print the output from the VL_1 AQ7 procedure.

6 - Print the selected meta functions in a table.

7,8 - 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 F1.

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 correspondes 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 acutally 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

VLMAXSTAR(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_2 formulas. There are four 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 F0 which intersect with this formula.

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

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(2) - The number of VL$_2$ 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 VL$_1$ variables when running in VL$_1$ 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 E 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.

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

F - 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. Data Structures

2.1 Constants

Some constant 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_1 mode, SYMSZE is the number of VL_1 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_1 type variables which are allowed in the program. This number being to 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 'stack overruns heap' 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 about 55, DEC is about 30).

LNK(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.

2.2 Parse table (PT)

The parse table consists of a data structure which represents the productions in the VL_2 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 in 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 corresponing row of the table is matched.

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

```
<VLRULE>                    ::= <NUMBER> <RULE> | <RULE>

<RULE>                      ::= <CONDITION> => <SELECTOR>

<CONDITION>                 ::= <CONDITION> <SELECTOR> | <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 0
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 and variable in the VL_2 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).

NARG - the number of arguments of a function.

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

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

EVAL - maximum value in complete domain.

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	MVAL	NVAL
FORALL	1	1	0	1	0	1	1	1
#PT	2	2	0	2	0	6	6	0
SHAPE	3	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	7	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 ontains 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 eleent 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 -F2COV (the secondary field). The columms are interpreted:

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

SYMPTR - 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 SYMPTR (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: (NMST=3)

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

with the three meta functions:

MS1 = #PT(SHAPE=0)
MS2 = FORALL(SHAPE=1)
MS3 = #PT(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 a 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.

FR - 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.

MSFL - a pointer to the meta selectors associated with the graph.

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).

VAL - the set of values associated with the node (for variables, 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.

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

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 0

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

2.7 VL_1 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 (AQPAR)

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_1 type selector in CVAL.

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

LOST - if true, then VL_1 compexes are stripped.

MAXSTARAQ - the maximum size of a partial star in AQ7

2.9 VL Parameters (PRM)

This structure contains parameters relevant to the VL_2 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}.

EXTMTY - a flag indicating whether EXTMTY 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 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

COVSET - pointer to the list of output formulas

STR,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. 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


  <blank line>

0 1 3 -3 0

1 2 -3 0

0 3 -4=> -6 0

0 4 -6 -4 0

1 5 -6 0

0 14[ -19= -10 ] 0

1 7[ -21( -14)= -10 ] 0

1 18[ -21( -14) ] 0

1 7[ -21= -10 ] 0

0 8 -20, -10 0

1 9 -20.. -20 0

1 19* 0

1 10 -20 0
```

```
0 11 -19, -14 0

1 20 -19. -14 0

1 12 -19 0

0 13 -19* -10; -17 0

1 14 -19= -10 0

0 15 2 0

0 16 3 0

0 17 1 0

0
```

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. Unfortunately, the numbers indicating which sets are to be covered may not be entered in this file (they must come from the terminal.)

3.4 VL1EVE

This file contains a list of VL_1 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 OFILE are the TTY input and output (these are TTY in the DEC 10 version). All other file 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 preform 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 MQ.

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 the data base. Find te first two numbers in the command (GETNUM) and place in the variables I and L. Then, execute the command.

PGRAPH - Print the gra structure as VL_2 formula. Assing indices to all varialbes. 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.

PCPY - Print in VL_1 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 carraige return from user.

4.2 VL Translation to Internal

TOKEN - Read an inut 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 Y in addition to a new row for X1). If CTYPE is 3, then comput 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_2 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 - Cooontains 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.

PSTK - the stack of arguments of a function (PSTK[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 altermative 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 in 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 FSTK, 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 FSTK[1] contains the location of the function.

4.3 VL_2 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). 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.

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 COVFR).

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

COVFR - 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 between nodes of G0 and nodes of G1 (i.e. correespondence 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

AQSET - Translate from VL_2 representation (graph structure) to VL_1 represenataion (sequence of sets of values). Create two sets of compxes, F1 containing subgraphs of graphs with VL_2 set F1, and F2, the set of complexes associated with c-structures (GSUB) isomorphisms with elements of the VL_2 set F2. The first element of F1 corresponds to the part of the graph GSUB which was consistent. The two sets of events are passed to the Q PROCEDURE WHICH RETURNS A COMPLEX COVERING THE FIRST ELEMENT OF F1 BUT NO ELEMENT OF F2. 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_1 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_1 variables, the remainder are nodes with COUNT = 1 in GSUB.

VL1 - Input VL_1 events from the file VL1EVE and translate to complex storage. Call AQ to find generalization and then print result.

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

COSTF - 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 ORDIPP = 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, count 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).

4.6 Supporting Routines

ILINE - input end of line from CFILE or the terminal

GETCHAR - read one character from the TTY or CFILE

PEOS - detect end of line on TTY or CFILE

INSIDE - determire if the set V_2 is a generalization of the set V_1

EXTND - find the extention 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).

LIST OF REFERENCES

1. Larson J., Inductive Inference in the Variable Valued Predicate Logic System VL_2: Methedology and Computer Implementation. Ph.D Thesis, Department of Computer Science, University of Illinois, 1977.

2. Larson J., Michalski R.S., "Inductive Inference of VL Decision Rules." Workshop on Pattern Directed Inference Systems, Hawaii 1977.

APPENDIX A

The file EXPLAIN

!1

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 PREVIOUS 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 REUIRED TO HAVE AT LEAST TWO ARGUMENTS.

SELECTORS ARE ADDED TO A PRODUCT C1 USING THE FOLLOWING ALGORITHM:

1 ALL VARIABLES (I.E. ARGUMENTS) WHICH ARE CONNECTED TO SELECTORS IN THE PORDUCT 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 EQUIVALAENT 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:

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 RE 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 THHE SET OF EVENT IN F1 AND F0. BELOW, THE C-FORMULA STRUCTURE AND INPUT EVNETS ARE LISTED. THE VL1 VARIABLES CORRESPOND 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 MANYSUCH 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.

!

!13

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

!

!100

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:

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 MQ 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 MQ LIST IS PRINTED BEFORE AQ GENERALIZATION AND THEN THE RESULTING FORMULA AFTER AQ GENERALIZATION IS PRINTED.

3 AFTER FULL GENERALIZATION, THE BEST MQ 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

7,8 NOT USED

9 PRINT ALL ALTERNATIVE GENERALIZATIONS OF THE EVENT

10 PRINT EVENT F1 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

AQCUTF1

IN ORDER TO SPEED UP THE AQ PROCEDURE, ONLY CUTF1 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

!

!300

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.

!

!400

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 WHIH IS CALCULATED AS FOLLOWS:

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:

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 LETTER OF THE PARAMETER NAME (E.Q. 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

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 ACTUALL 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 IN THE VL1 SET 1

6 THE NUMBER OF EVENTS COVERED IN THE VL1 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: VCOST(SHAPE]=-2 SETS THE COST OF THE DESCRIPTOR SHAPE TO -2.

!

!800

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.

!

!900

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, OTHER WISE IT IS RELATIVE TO THE MAXIMUM AND MINIMUM COSTS IN THE PARTIAL STAR. THE VALUE IS ENTERED IN HUNDRETHS (SEE AQTOLERANCE).

EXAMPLE: VLTOL(3)=200 SPECIFIES THAT THE THIRD VL2 COST CRITERION (VLCRIT(2)) HAS AN ABSOLUTE TOLERANCE OF 2 (=2.00)

!

!1000

VLCRIT

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

1 THE NEGATIVE OF THE NUMBER OF EVENTS OF F1 COVERED BY THIS C-FORMULA BUT NOT BY ANY PREVIOUS LQ

2 THE NUMBER OF SELECTORS IN THE C-FORMULA.

3 THE NUMBER OF EVENTS IN F0 COVERED BY THE C-FORMULA

4 THE TOTAL SUM COST OF VARIABLES IN SELECTORS. IF A FUNCTION APPEARS MORE THAN ONCE IN THE FORMULA, THEN IT IS COUNTED FOR EACH APPEARANCE, NOT JUST ONCE.

THIS PARAMETER IS SPECIFIED BY ENTERING

VLCRIT(I)=J WHICH SPECIFIES THAT THE ITH CRITERION IS NUMBER J ABOVE.

EXAMPLE: VLCRIT(1)=3

!

!1100

VLNF

THIS PARAMETER SPECIFIES THE NUMBER OF COST CRITERIA WHICH ARE TO BE USED IN THE VL2 TRIMMING AND SELECTION PROCESS.

!

!1200

NCONSIST

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

!

!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 OF 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.

!

!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.

!

!1500

```
        NOPRULE
        THIS  PARAMETER  TURNS  OFF  THE  PRINTING  OF  RULES.   SEE
PRULE.
        !
!1600
        NOTRACE
        THIS  PARAMETERS  ALLOWS  THE  USER  TO  TURN  OFF  A  TRACE
FEATURE (SEE TRACE) TO TURN OFF A TRACE FEATURE I, ENTER
        NOTRACE I
        !
!1700
        QUIT
        RETURN  TO THE COMMAND 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  PROGRAM   AT  A   PARTICULAR  TRACE   FEATURE.
GENERALLY,  THIS  MAY  BE  USED  TO  GET  AN  EXPLANATION  OF  WHATS
HAPPENING OR TO CHANGE SOME PARAMETER.
        !
!2100
        NOSTP
        TURN  OFF THE  STOP IN A  TRACE.   TO TURN OFF  THE STOP FOR
TRACE FEATURE I ENTER
        NOSTP I
        !
```

```
!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
        !
!1400
        PRINT
        THIS  PARAMETER  REQUESTS  A  LIST  OF  THE  META  SELECTORS
CURRENTLY  SELECTED,  THE  DMAIN  STRUCTURES,  THE  INPUT  RULES  OR
RESTRICTIONS. ENTER:
        PRINT M FOR META SELECTORS
        PRINT D FOR DOMAINS
        PRINT R FOR RESTRICTIONS
        PRINT F FOR INPUT DECISION RULES.
        !
!1500
        METATRIM
```

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

!

!2800

LQST

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

!10

AN EVENT E1 OF F1 HAS BEE 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 F1 IS FOUND, ALL EVENTS COVERED BY THIS LQ ARE REMVED 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 WYMBOL AND ARGUMENTS WHERE EACH ARGUMENT APPEARS IN THE CONITION PART OF THE RULE SOMEWHERE.

EXAMPLE

```
        [LEFT(X1,X2)][LEFT(X2,X3)]=>[LEFT(X1,X3)].
        [STA(Y1)=1][PART(Y1,L1)]=>[COND(L1)=*].
        !
!??
        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:
        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  OFCOMPARABLE  DUMY  VARIABLES)  AND  A  NUMBER  WHICH
DISTINGUISHES  THIS ARGUMENT  FROM OTHERS OF  THE SAME  GROUP (E.G.
Y1  OR  CAP4).  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
HAS AN INTERVAL DOMAIN STRUCTURE).
       SELECTOR   EXAMPLES:   [SH(X1)=1,2]   [P(X1,X2)]  [SH(A1)=*]
[SIZE(L1)=1..6]
       RULE EXAMPLE: [SH(X1)=3][Q(X1,X2)]=>[D=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 GENERALIZTION RULE.
       EXAMPLE: [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 MODE
```

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 FROM A FILE VL1EVE. THE FORMAT OF THIS FILE CONTAINS A LIST OF EVENTS (VALUES OF VARIABLES) PRECEDED BY THE DECISION VALUE. FOR EXAMPLE, IF THERE ARE TWO EVENTS IN SET 1 AND 2 EVENTS IN SET 5, THEN ENTER INTO THE FILE:

```
1 3 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 PROGRAM AND ENTER V. AT THIS POINT, YOU MAY ENTER DOMAIN STRUCTURES (IN THE VL2 FORMAT), ENTER PARAMETERS (THIS ALLOWS ONE TO ENTER COST FUNTIONS 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 F COMMAND. THE DOMAIN TYPES ARE:

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 PRECIDATES LIKE LST- AND MST-
        !
!29
        S - EQUIV PREDICATES
        ADD EQUIVALENCE TYPE PREDICATES (E.G [SH(X1.X2)=SAME])
        !
```

APPENDIX B

The BOSS file which converts from CYBER to DEC

```
VS/SEGMENTED//W
VS/$/:/W
VS/READ(IFILE/READ(TTY/W
VS/WRITE(OFILE/WRITE(TTY/W
VS/GETSEG(IFILE/READLN(TTY/W
VS/PUTSEG(OFILE)/BREAK/W
VS/WRITELN(OFILE/WRITELN(TTY/W
VP1
VF/PROGRAM VL2/
M/(* /;S/;/*)/
VS/<>/ /W
VS/EOS(IFILE)/EOLN(TTY)/W
VP1
<VLF/LABEL/?;VM/(* /;VS/;/;*)/.>
VS$*)$*/$W
TAB ?
VS$(*$/*$W
P1
<LVF1$/*$?;VS$/*$ /*$.>
TAB 80
I-1/"*ID SYSTEM=PRINT,PRINT=DEC10,NAME='VL2.PAS(4113,1374)'
S*"*/*
READW -1=JCL
POP
VS/+PREM/ OR PREM/W
VS/TRSLT+/TRSLT OR /W
VS/*TRSLT/ AND TRSLT/W
VS/+[/ OR [/W
VS/+ [/ OR [/W
VS/V1 */V1 AND/W
VS/CVAL[ I ]*/CVAL[ I ] AND /W
VS/CVAL[ I ]+/CVAL[ I ] OR /W
VS/EOLN(IFILE/EOLN(TTY/W
```

APPENDIX C
PROGRAM LISTING

```
(*VL2*)
(*$D+
                       VL2-SYNTHESIS OF VL2 FORMULAS
    THIS PROGRAM SYSTHEIZES VL2 FORMULAS (REPRESENTED AS DECISION RULES)
WHICH ARE GENERALIZATIONS OF A SET OF OF VL2 FORMULAS.  ASSUMTIONS ARE
THE FOLLOWING:
    1. ALL VARIABLES ARE EXISTENTIALLY QUANTIFIED AND REPRESENT
    DISTINCT VALUES OF THEIR DOMAIN.
    2. EACH EXPRESSION IS ASSUMED TO BE A PRODUCT OF SELECTORS IN VL2
    WITH ATOMIC FORMS WHICH ARE FUNCTINS OF SIMPLE VARIABLES
    3. EXPRESSIONS ARE REQUIRED TO BE IN A FORM WHICH CAN BE TRANSLATED
    INTO A CONNECTED GRAPH.  MORE PREDICATES MAY BE ADDED BY THE USER TO ASSURE
    THIS.
        THE PROGRAM GENERATES LARGER AND LARGER PRODUCTS OF SELECTORS
    WHICH COVER A SPECIFIC ELEMENT OF THE SET OF FORMULAS WHICH ARE
    TO BE COVERED.  WHEN ONE PRODUCT IS FOUND WHICH DOESNT COVER
    ANY FORMULA IN OTHER SETS, AN AQVAL/1 TYPE PROCEDURE IS CALLED
    TO EXTND THE REFERENCES.  COVERING IS TESTED BY A SUBGRAPH
    MATCHING ALGORITHM WHICH FINDS A SPANNING TREE OF THE SMALLER
    OF THE TWO GRAPHS AND TRIES TO FIND A TREE IF THE LARGER
    GRAPH WHICH MATCHES.  A BACKTRACK MECHANISM IS BUILT IN TO
    TO BACK DOWN THE TREE IF SOME MATCH FAILS.
       ANY DESCRIPTOR FOLLOWED BY A NUMBER IS A DUMMY VARIABLE.
USING VL2 ON THE CYBER
IOFILES:
    THERE ARE SEVERAL FILES WHICH THE PROGRAM USES.  THEY ARE BRIEFLY DESCRIBED
BELOW:
IFILE - INPUT FROM TTY
OFILE - OUTPUT TO TTY
OUTPUT - OUTPUT ERROR MESSAGES AND DEBUG OUTPUT
STAB - SYMBOL TABLE - TO START ON A NEW PROBLEM, THIS FILE MAY BE EMPTY.
    THE PROGRAM LOADS DESCRIPTORS INTO THIS FILE AT THE END OF EACH
    SESSION (Q-COMMAND).
TABLES - PARSE TABLE WHICH CONTAINS VALID SYNTAX OF VL2 EXPRESSIONS
GFILE - STORAGE OF INTERNAL RULE FORMAT.  THE Q-COMMAND AUTOMATICALLY
    STORES RULES INTO GFILE AND THE SYMBOL TABLE INTO STAB.  WHEN THE
    PROGRAM IS RERUN, THE RULES AND STAB ARE READ BACK INTO CORE
CFILE - OPTIONAL COMMAND FILE.  IF COMMANDS ARE HERE, THE FIRST LINE MUST
    BE BLANK FOLLOWED BY LINES OF INPUT A ONE WOULD ENTER ON THE TERMINAL
RUNNING THE PROGRAM
   TYPE :  V,,INPUT,OUTPUT        <V - OBJECT CODE FOR PROGRAM>
          <ENTER CARRAIGE RETURN AT ?>
BUILDING THE RULE BASE
    SCRATCH STAB AND GFILE (RETURN,STAB,GFILE).
    BUILD A COMMAND FILE (CFILE).
        USE COMMANDS M AND A AS FOLLOWS:
            <BLANK>
            M     <COMMAND TO MODIFY RULE BASE>
            A     <COMMAND TO ADD RULE TO BASE>
            [SH(X1)=1][SH(X2)=1][P(X1,X2)=1]    <PART OF RULE>
            ->[D=1].                            <CONSEQUENCE>
            M
            A
            [SH(X1)=2][SH(X2)=1][P(X1,X2)=1]->[D=2].
        ALWAYS TERMINATE A RULE WITH A PERIOD.  PREMISE SHOULD FORM A
        CONNECTED C-GRAPH (CONJUNCTIVE GRAPH) WHEN TRANSLATED, CONSEQ
        SHOULD BE A SELECTOR WITHOUT ARGUMENTS.
    TO ENTER THE RULES INTO THE RULE BASE, RUN THE PROGRAM
    AND ENTER THE COMMAND (R).
    EXAMINING OR DELETING RULES
    AFTER BUILDING THE RULE BASE, RUN THE PROGRAM AND ENTER
    THE COMMAND M FOLLOWED WITH D.  THE PROGRAM ASKS WHICH SET YOU
    WANT TO LOOK AT, ENTER THE SET (1 TO 5).  IN RESPONSE TO
    THE COMMAND DELETE RULE, ENTER Y (DELETE THE RULE JUST PRINTED
    OUT), N (DONT DELETE THIS RULE) OR Q (RETURN TO COMMAND LEVEL).
    CHANGE PARAMETERS
       ENTER THE P COMMAND AND THEN THE PARAMETERS WHEN ASKED.
    ADD DOMAIN STRUCTURES
        ENTER E COMMAND AND THEN THE STRUCTURE.  THESE STRUCTURES ARE
    NOT CURRENTLY STORED FROM ONE EXECUTION TO THE NEXT.  ENTER
   THE STRUCTURE AS FOLLOWS:
             E    <E COMMAND>
             [SH=1,2,3,5]->[SH=10].
        NOTE THAT THE DESCRIPTORS ARE GIVEN WITHOUT ARGUMENTS AND THAT
   ELEMENTS IN THE REFERENCE ARE SEPARATED BY COMMAS.  THE ENTIRE
   RULE IS TERMINATED WITH A PERIOD.
   COVER SET OF RULES
        ENTER THE C COMMAND AND THEN THE SET WHICH IS TO BE COVERED.
   THE PROGRAM PRINTS OUT INTERMEDIATE RESULTS:
   1. EACH CONSISTENT FORMULA IS PRINTED AS IT IS FOUND
   2. IF IT IS NOT ALREADY IN THE STAR, THEN THE GENERALIZATION
   OF THE FORMULA IS PRINTED ALONG WITH STEPS IN THE GENERALIZATION
   PROCESS
   3. THE RULE WHICH IS SELECTED IS PRINTED AND ALL FORMULAS
   WHICH ARE COVERED BY THIS FORMULA ARE LISTED.*)
 PROGRAM VL2(OUTPUT,IFILE,OFILE,STAB,GFILE,TABLES,CFILE,EXPLAIN,VL1EVE);
 LABEL 1,2,3,4,5,99;
CONST
  SYMSZE = 36;(*' OF DESCRIPTORS +' OF DUMMY VARIABLES +10 => ' ROWS IN STAB*)
  NDES = 15;  (*NUMBER OF ENTRIES IN DSTRUC RECORD *)
  GSIZE = 36;(*' OF DUMMY VBLS + ' SELECTORS IN AN EVENT + 10 => ' NODES IN G*)
  MNVAL = 15;  (* MAXIMUM NUMBER OF VALUES IN DOMAIN*)
  MLNK = 18;(* MAXIMUM ' OF LINKS TO ANY NODE +1*)
TYPE
  PT = RECORD
     RHS : ARRAY[1..21,1..13] OF INTEGER;
     CONT : ARRAY[1..21] OF BOOLEAN;
     SRULE: ARRAY[1..21] OF INTEGER
     END;
  VALTP = SET OF 0..MNVAL;
  NODEA = PACKED ARRAY[1..MLNK] OF 0..GSIZE;(*TYPE FOR NODE LIST*)
  CPX = RECORD
     COST : INTEGER;  (* COST OF COMPLEX *)
     FQ : BOOLEAN;  (* LIST OF COMPLEXES NOT COVERED BY ANY LQ *)
     FP : BOOLEAN;(* LIST OF COMPLEXES NOT COVERED BY ANY STAR *)
     CVAL :  PACKED ARRAY[1..GSIZE] OF VALTP;(* SELECTOR VALUES *)
     NXTC : CPX  (* POINTER TO NEXT COMPLEX *)
     END;
  GRAPH = RECORD
     COEF : INTEGER;
     RNO : INTEGER;  (*RULE NUNBER*)
     FP : BOOLEAN;  (*TEMPORARY FLAG USED IN COVER PROCEDURE*)
     MSEL : CPX;
     COST : ARRAY[1..4] OF INTEGER;(*COST OF THIS FORMULA*)
     ESET : VALTP;
     VBL: PACKED ARRAY[1..GSIZE] OF BOOLEAN;  (* TRUE IF ENTRY IS DUMMY *)
     ORDIFF : PACKED ARRAY[1..GSIZE] OF BOOLEAN;
(* TRUE IF ORDER OF ARGS IRRE*)
     VAL : PACKED ARRAY[1..GSIZE] OF VALTP;  (* VALUE OF THIS NODE *)
     COUNT : PACKED ARRAY[1..GSIZE] OF INTEGER;  (* NO OF TIMES USED IN NEWG *)
     ASSGN : PACKED ARRAY[1..GSIZE] OF 0..GSIZE;  (* ASSIGNMENT OF NODE *)
     PNO : PACKED ARRAY[1..GSIZE] OF -SYMSZE..SYMSZE;  (* DESC NUMBER *)
     DUMNUM : PACKED ARRAY[1..GSIZE] OF 0..SYMSZE;  (* WORK PACKED ARRAY *)
     NXTN : GRAPH;  (* POINTER TO NEXT GRAPH *)
     NNEG : GRAPH;  (* POINTER TO NEG GRAPH *)
     LNK : ARRAY[1..GSIZE] OF NODEA (*LINKS FOR NODES*)
     END;
  SYMTAB = RECORD
     NELT : INTEGER;
     NAME : PACKED ARRAY[1..SYMSZE,1..10] OF CHAR;  (* NAMES OF DESC *)
     PNO : ARRAY[1..SYMSZE] OF INTEGER;  (* DESC NO *)
     DPNO : ARRAY[1..SYMSZE] OF INTEGER;  (* DESC NO OF ASSOC DESC *)
     NARG : ARRAY[-SYMSZE..SYMSZE] OF INTEGER;  (* NUMBER OF ARGS *)
     VTYPE : ARRAY[1..SYMSZE] OF 1..3;  (*TYPE OF VAR - 1-NOMINAL,2-INT,3-STRU*)
     VCOST : ARRAY[-SYMSZE..SYMSZE] OF INTEGER;(*COST OF EACH VARIABLE*)
     EVAL : ARRAY[1..SYMSZE] OF INTEGER;  (* NUMBER OF VALUES IN EXTND DOM*)
     MVAL : ARRAY[1..SYMSZE] OF INTEGER;  (* MINIMUM VALUE OF REF *)
    NVAL : ARRAY[1..SYMSZE] OF INTEGER  (* NUMBER OF VALUES *)
     END;
  MSTR = RECORD
     PNO : PACKED ARRAY [1..GSIZE] OF 0..SYMSZE;
```

```
      VAL : PACKED ARRAY [1..GSIZE] OF 0..MNVAL;
      SYMPTR : PACKED ARRAY [1..GSIZE] OF 0..SYMSZE;
      PTR : PACKED ARRAY[1..GSIZE] OF 0..GSIZE;
      F1COV : PACKED ARRAY [1..GSIZE] OF INTEGER;
      F2COV : PACKED ARRAY[1..GSIZE] OF INTEGER;
      METATRIM : INTEGER;
      NMST : INTEGER
       END;
   AQPARM = RECORD
      NVAR : INTEGER; (* NUMBER OF VARIABLES IN AQ PROC *)
      CSTF : ARRAY[1..6] OF INTEGER; (* COST FUNCTION A LIST *)
     TOLER : ARRAY[1..6] OF REAL; (* TOLERANCE LIST *)
      NF : INTEGER;(* NUMBER OF COST FORMULAS TO BE USED *)
      FRFEC : CPX; (* POINTER TO FREE COMPLEX LIST *)
      SLOC : ARRAY[1..GSIZE] OF INTEGER; (* LOCATION IN THE STABLE OF VBL*)
      CUTF1 : INTEGER;(* NUMBER OF F1 TO CHECK IN AQ *)
        LOST:BOOLEAN;
      MAXSTARAQ : INTEGER (* MAXSTAR PARM IN AQ ALG *)
      END;
PARM = RECORD
      CSTF : ARRAY[1..6] OF INTEGER;(*VL2 COST FUNCTIONS*)
      TOLER : ARRAY[1..6] OF REAL;(*VL2 TOLERANCE*)
      NF : INTEGER;(*NUMBER OF COST FUNCTIONS*)
      MAXSTAR : INTEGER;(*MAXSTAR PARAMETER*)
      ALTER : INTEGER; (* NUMBER OF ALTERNATIVES *)
      EXTMTY:BOOLEAN;
      EQUIV : BOOLEAN;
      NCONSIST : INTEGER (* NUMBER OF CONSISTENT ALTERNS TO GENERATE*)
      END;
  CARRAY = ARRAY[1..101] OF CHAR;
  IARRAY = ARRAY[0..MNVAL] OF INTEGER;
  DSTRUC = RECORD
   PREM : ARRAY[1..NDES] OF VALTP; (* PREMISE OF DESC STRUCTURE RULE *)
   CONS : ARRAY[1..NDES] OF VALTP; (* CONSEQUENCE OF DESC STRUCTURE RULE *)
   PNO : ARRAY[1..NDES] OF INTEGER; (* POINTER TO SYMBOLTABLE *)
   NELE : INTEGER (* NUMBER OF ELEMENTS IN THIS STRUCTURE USED SO FAR*)
      END;
   GPTR = GRAPH;
   DPTR = DSTRUC;
   PPTR = PT;
   SPTR = SYMTAB;
   APTR = AQPARM;
   CPTR = CPX;
   GSAR = ARRAY[1..5] OF GPTR;
   VAR CHRR,CHRR1:CHAR;I,J,K,ES,ERR,NINSTR,INFILE,NMQ:INTEGER;
     (* ES - INDEX OF DECISION WHICH IS BEING COVERED
        NINSTR - NUMBER OF G STRUCT IN VL2 STAR
        CURRENT INPUT FILE (0 - TTY, 1 - CFILE*)
       DST : DSTRUC;
        FREEG,G1,G2,G,STAR,RESTLIST,R:GPTR;
       COVSET,GSET,MQ,PSTAR,OPSTAR:GPTR;
       MST : MSTP;
     (*GSET - POINTERS TO LIST OF C GRAPHS FOR EACH DECISION
      RESTLIST - POINTER TO LIST OF RESTRICTIONS
        CSET - POINTERS TO LIST OF IRRED GENERALIZATIONS
        FREEG - POINTER TO LIST OF UNUSED G STRUCT
        STAR - POINTER TO CURRENT VL2 STAR*)
       NF1,CRULENO,NEWTRACE,NTIMES:INTEGER;PTBL:PT;S:SYMTAB;
    STP,TRACE : SET OF 1..10;PRULE : BOOLEAN;
         FIXIT : VALTP;
       AQP:AQPARM;
      CNSTCY : ARRAY[1..GSIZE] OF INTEGER; (*CONSISTENCY VALUES*)
     PRM : PARM;
    STAB : FILE OF SYMTAB;
   VL1EVE : FILE OF CHAR;
  IFILE : SEGMENTED FILE OF CHAR;
  OFILE :  SEGMENTED FILE OF CHAR;
 GFILE : FILE OF GRAPH;
    TABLES : FILE OF CHAR;
    CFILE : FILE OF CHAR;
     EXPLAIN : FILE OF CHAR;
```

```
      DFILE : FILE OF  DSTRUC;
 (*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                              PGRAPH(G:GPTR;S:SYMTAB);FORWARD;
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
 PROCEDURE PGRAPH(G:GPTR;S:SYMTAB);FORWARD;
PROCEDURE ENTERP;FORWARD;
PROCEDURE VLINT(G:GPTR;VAR ERR:INTEGER;VAR ES:INTEGER);
   FORWARD;
 (*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                              INSIDE(DNUM:INTEGER;V1,V2:VALTP):BOOLEAN;
       DETERMINES IF ONE SET, V2 IS A GENERALIZATION OF THE SET V1.  IF EVAL A
   AND NVAL ARE THE SAME, THEN THE DOMAIN IS ASSUMED TO BE STRUCTURED
   OTHERWISE, IT IS CARTESIAN.  IF STRUCTURED, THEN THE STRUCTURE DSTRUC IS
   SEARCHED FOR POSSIBLE GENERALIZATIONS.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
 FUNCTION INSIDE(DNUM:INTEGER;V1,V2:VALTP;INSD:BOOLEAN):BOOLEAN;
 VAR I,J:INTEGER;
  BEGIN
   INSIDE:=FALSE;
   DNUM:=ABS(DNUM);
   IF S.VTYPE[DNUM]<>3 THEN
     IF INSD AND (V1<=V2) OR (NOT INSD AND (V1 * V2 <>[ ])) THEN
       INSIDE:=TRUE
       ELSE
       ELSE
       WITH DST DO
         BEGIN
           FOR I:=NELE DOWNTO 1 DO
             IF DNUM=PNO[I] THEN
               IF CONS[I]<=V2 THEN
                 V2:=V2+PREM[I];
           IF INSD AND (V1<=V2) OR (NOT INSD AND (V1 * V2 <>[ ])) THEN
             INSIDE:=TRUE;
           END;
     (*WITH*)
   END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                              ADDSEL(G:GPTR);
       ADD SYMMETRIC SELECTORS TO THE G STRUCT.  FIND ALL SELECTORS WHICH
   INVOLVE THE SAME FUNCTION AND SAME REFERENCE.  FORM A NEW SELECTOR
   WHICH IS LINKED TO ALL THESE.  THE PNO OF THE NEW SELECTOR
   IS THE NEGATIVE OF THE PNO OF THE ORIGINAL SELETORS.  VBL IS
   SET TO FALSE IN THESE SELECTORS AND ORDIER IS ALSO SET TO TRUE.
   THEREFORE, SUBSETS OF ARGUMENTS CAN BE COMPARED USING SUBG1.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
   (*INSIDE*)
  PROCEDURE ADDSEL(G:GPTR);
  VAR LND,NND,I,J,K,L:INTEGER;
  BEGIN
   WITH G DO
     BEGIN
       LND:=1;
       WHILE LNK[LND,1]<>0 DO
         LND:=LND+1;
       NND:=LND-1;
       FOR I:=1 TO NND DO
         COUNT[I]:=0;
       FOR I:=1 TO NND DO
         IF (COUNT[I]=0)AND(NOT VBL[I]) AND(LNK[I,2]=0) THEN
           BEGIN
             K:=2;
             LNK[LND,1]:=LNK[I,1];
             FOR J:=I+1 TO NND DO
               IF (VAL[J]=VAL[I])AND(PNO[I]=PNO[J]) THEN
                 BEGIN
                   COUNT[J]:=1;
                   LNK[LND,K]:=LNK[J,1];
                   K:=K+1;
                   END;
             LNK[LND,K]:=0;
             IF K<>2 THEN
```

```
                BEGIN
                  PNO[LND]:=-PNO[I];
                  S.VCOST[PNO[I]]:=S.VCOST[-PNO[I]];
                            (* USED TO SELECT EQUIV TYPE SELECTORS IN COVER SO
                            RT*)
                  VAL[LND]:=[0..MNVAL];
                  VBL[LND]:=FALSE;
                  ORDIRR[LND]:=TRUE;
                  FOR J:=1 TO K-1 DO
                    BEGIN
                                  (*ADD BACK POINTERS*)
                      L:=1;
                      WHILE LNK[LNK[LND,J],L]<>0 DO
                        L:=L+1;
                      LNK[LNK[LND,J],L]:=LND;
                      LNK[LNK[LND,J],L+1]:=0;
                      END;
                            (*FOR J*)
                  LND:=LND+1;
                  END
                      (*K<>2*)
              ELSE
              LNK[LND,1]:=0;
              END;
            (*---AND---*)
        END;
      (*WHILE*)
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                               EXTND(DNUM:INTEGER;V1,V2:VALTP);
         FIND THE EXTENSION OF V1 AGAINST V2, PUT RESULT IN RSLT.  DNUM IS THE
    DESCRIPTOR NUMBER (LOC IN STAB).  IF EVAL = NVAL, THEN THE DOMAIN
    IS ASSUMED TO BE CARTESIAN, OTHERWISE, THE DSTRUC RECORDS ARE
    SEARCHED TO FIND THE GENERALIZATION.  DSTRUC RECORDS ARE ASSUMED
    TO BE IN THE ORDER: LOWEST LEVEL GENERALIZATION FIRST.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
   (*ADDSEL*)
  PROCEDURE EXTND(DNUM:INTEGER;
  V1,V2:VALTP);
  VAR I,J,LL:INTEGER;
  TRSLT:VALTP;
  BEGIN
    TRSLT:=V1;
    DNUM:=ABS(DNUM);
    CASE S.VTYPE[DNUM] OF
1:      TRSLT:=[0..MNVAL]-V2;
2:      BEGIN
          I:=0;
          WHILE (I<MNVAL)AND(NOT (I IN V1)) DO
            I:=I+1;
          J:=0;
          WHILE (J<MNVAL)AND(NOT (J IN V2)) DO
            J:=J+1;
          IF I<J THEN
            FOR LL:=I TO J-1 DO
              TRSLT:=TRSLT+ [LL]
              ELSE
              BEGIN
                WHILE (J<MNVAL)AND(J IN V2) DO
                  J:=J+1;
                FOR LL:=J TO MNVAL DO
                  TRSLT:=TRSLT+ [LL];
                END;
          END;
           (*CASE 2*)
3:      WITH DST DO
          BEGIN
            FOR I:=1 TO NELE DO
              IF DNUM=PNO[I] THEN
                IF V1<=PREM[I] THEN
                  IF NOT INSIDE(DNUM,V2,CONS[I],TRUE) THEN
                  V1:=CONS[I];
          TRSLT:=V1;
          FOR I:=NELE DOWNTO 1 DO
            IF DNUM=PNO[I] THEN
              IF CONS[I]<=TRSLT THEN
                TRSLT:=TRSLT+PREM[I];
          END
         (*WITH*)
      END;
       (*CASE STMT*)
    FIXIT :=FIXIT*TRSLT;
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                               ILINE;
    INPUT LINE OF INFORMATION FROM TTY OR CFILE DEPENDING ON INFILE
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
   (*EXTND*)
  PROCEDURE ILINE;
  LABEL 1;
  BEGIN
    IF INFILE=0 THEN
1:    BEGIN
        GETSEG(IFILE);
        WHILE EOLN(IFILE) DO
          GETSEG(IFILE);
        END
    ELSE
    BEGIN
      IF EOF(CFILE) THEN
        BEGIN
          INFILE:=0;
          GOTO 1;
          END;
      READLN(CFILE);
      IF EOF(CFILE) THEN
        BEGIN
          INFILE:=0;
          GOTO 1;
          END;
      END;
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                               GETCHRR(VAR C:CHAR);
         GET CHARACTER FROM INPUT FILE
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
   (*ILINE*)
  PROCEDURE GETCHRR(VAR C:CHAR);
  BEGIN
    IF INFILE=0 THEN
      READ(IFILE,C)
      ELSE
      READ(CFILE,C);
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                               PEOS(I:INTEGER):BOOLEAN;
         DETERMINE IF AT THE END OF SEGMENT OR LINE
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
  FUNCTION PEOS(I:INTEGER):BOOLEAN;
  BEGIN
    PEOS:=FALSE;
    IF INFILE=0 THEN
      IF EOLN(IFILE) THEN
        PEOS:=TRUE
        ELSE
        ELSE
        IF EOLN(CFILE) THEN
          PEOS:=TRUE;
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                               INIT(
         INTIALIZE CERTAIN PARAMETERS, READ IN SYMBOL TABLE AND PARSE TABLE
    FROM STAB AND TABLES.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
  PROCEDURE INIT;
  VAR I,J:INTEGER;
  BEGIN
    TRACE:=[ ];
    PRULE:=TRUE;
    STP:=[ ];
    GSET:=NIL;
    FREEG:=NIL;
    RESTLIST:=NIL;
    DST.NELE:=0;
    MST.NMST:=0;
    MST.METATRIM:=3;
    FOR I:=1 TO GSIZE DO
      MST.PTR[ I ]:=I;
    INFILE:=0;
    RESET(TABLES);
    FOR I:=1 TO 6 DO
      BEGIN
        AQP.CSTF[ I ]:=I;
        PRM.CSTF[ I ]:=I;
        PRM.TOLER[ I ]:=0.0;
        PRM.NF:=3;
        PRM.NCONSIST:=4;
        AQP.TOLER[ I ]:=0;
        END;
    PRM.CSTF[ 1 ]:=3;
    PRM.CSTF[ 2 ]:=-1;
    AQP.CSTF[ 1 ]:=-1;
    AQP.CSTF[ 5 ]:=-5;
    AQP.CSTF[ 3 ]:=4;
    AQP.CSTF[ 4 ]:=3;
    PRM.CSTF[ 3 ]:=2;
    PRM.TOLER[ 1 ]:=0.3;
    AQP.NF:=2;
    STAR:=NIL;
    PSTAR:=NIL;
    GSET:=NIL;
    COVSET:=NIL;
    PRM.MAXSTAR:=2;
    AQP.MAXSTARAQ:=2;
    AQP.LQST:=TRUE;
    PRM.EXTMTY:=FALSE;
    PRM.EQUIV:=FALSE;
    AQP.CUTF1:=20;
    NEWTRACE:=0;
    AQP.FREEC:=NIL;
    AQP.NVAR:=0;
    CRULENO:=1;
    PRM.ALTER:=2;
    FOR I:= 1 TO 21 DO
      BEGIN
        READLN(TABLES);
        READ(TABLES,J);
        IF J=1 THEN
          PTBL.CONT[ I ]:=TRUE
          ELSE
          PTBL.CONT[ I ]:=FALSE;
        READ(TABLES,PTBL.SRULE[ I ]);
        J:=1;
        REPEAT
          READ(TABLES,CHRR);
          IF CHRR<>' ' THEN
            PTBL.RHS[ I,J ]:=ORD(CHRR)
            ELSE
            READ(TABLES,PTBL.RHS[ I,J ]);
          J:=J+1;
          UNTIL PTBL.RHS[ I,J-1 ]=0;
        END;
      (*FOR I:= *)
    RESET(STAB);
    S.NELT:=0;
    FOR I:=1 TO SYMSZE DO
      BEGIN
        S.NVAL[I]:=0;
        S.VTYPE[I]:=1;
        S.EVAL[I]:=0;
        S.VCOST[I]:=0;
        S.MVAL[I]:=MNVAL;
        FOR J:=1 TO 10 DO
          S.NAME[I,J]:=' ';
        END;
    S. NAME[S.NELT+1]:='FORALL    ';
    S. NAME[S.NELT+2]:=' PT       ';
    S. NELT:=S.NELT+2;
    S. PNO[S.NELT-1]:=S.NELT-1;
    S. PNO[S.NELT]:=S.NELT;
    S. VTYPE[S.NELT]:=2;
    S. MVAL[S.NELT]:=0;
    S. NVAL[S.NELT]:=0;
    S. EVAL[S.NELT]:=0;
    S. MVAL[S.NELT-1]:=0;
    S. NVAL[S.NELT-1]:=1;
    S. EVAL[S.NELT-1]:=1;
    IF NOT EOF(STAB) THEN
      S:=STAB;
    RESET(DFILE);
    IF NOT EOF(DFILE) THEN
      DST:=DFILE;
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                            NEWG
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
   (* INIT *)
  PROCEDURE NEWG(VAR G:GPTR);
  BEGIN
    G:=FREEG;
    IF FREEG=NIL THEN
      NEW(G)
      ELSE
      FREEG:=G.NXTN;
    G. FP:=TRUE;
    G. RNO:=CRULENO;
    G. MSEL:=NIL;
    CRULENO:=CRULENO+1;
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                               GIN(G:GPTR);
         INPUT GRAPH STRUCTURE
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
  PROCEDURE GIN(G:GPTR);
  BEGIN
    G:=GFILE;
    GET(GFILE);
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                               GOUT(G:GPTR);
         OUTPUT GRAPH STRUCTURE
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
  PROCEDURE GOUT(G:GPTR);
  BEGIN
    GFILE:=G;
    PUT(GFILE);
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                              EXPLN
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
  PROCEDURE EXPLN(I:INTEGER);
  LABEL 11;
  VAR CHRR:CHAR;
  PROCEDURE RDEX(I:INTEGER);
  LABEL 99;
  VAR J:INTEGER;
  BEGIN
    RESET(EXPLAIN);
    CHRR:=' ';
    J:=-1;
    WHILE J<>I DO
      BEGIN
        WHILE CHRR<>'!' DO
          BEGIN
            READLN(EXPLAIN);
            IF EOF(EXPLAIN) THEN
              BEGIN
                WRITELN(OFILE,'NO HELP');
                PUTSEG(OFILE);
                GOTO 99;
                END;
            READ(EXPLAIN,CHRR);
            END;
        READ(EXPLAIN,J);
        CHRR:=' ';
        END;
    WRITELN(OFILE);
    WRITELN(OFILE);
    REPEAT
      READLN(EXPLAIN);
      WRITELN(OFILE);
      WHILE NOT EOLN(EXPLAIN) DO
        BEGIN
          READ(EXPLAIN,CHRR);
          WRITE(OFILE,CHRR);
          END;
      IF CHRR='*' THEN
        BEGIN
          WRITELN(OFILE);
          WRITE(OFILE,'PRESS RETURN TO CONTINUE');
          PUTSEG(OFILE);
          GETSEG(IFILE);
          END;
      UNTIL CHRR='!';
    WRITELN(OFILE);
    WRITELN(OFILE);
    WRITELN(OFILE);
99: END;
   (* RDEX*)
  BEGIN
    IF (0<=I)AND(I<=10) THEN
      IF I IN STP THEN
        BEGIN
          WRITELN(OFILE);
          WRITELN(OFILE, 'STOP AT TRACE LEVEL',I:2);
11:       WRITELN(OFILE,'ENTER ? FOR EXPLANATION',
' P TO CHANGE', ' PARAMETERS OR RETURN TO CONTINUE');
          PUTSEG(OFILE);
          GETSEG(IFILE);
          IF NOT EOLN(IFILE) THEN
            BEGIN
              READ(IFILE,CHRR);
              IF CHRR='P' THEN
                ENTERP
                ELSE
                BEGIN
                  RDEX(I);
                  GOTO 11;
                  END;
              END;
          END
    ELSE
    ELSE
    RDEX(I);
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                             PRINT METAD
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
   (*EXPLN*)
  PROCEDURE PMETAD;
  VAR I:INTEGER;
  BEGIN
    WRITELN(OFILE,'THE SELECTED META-SELECTORS ARE:');
    WRITELN(OFILE,' MS TYPE          FUNCTION        F1COV FOCOV');
    FOR I:=1 TO MST.NMST DO
      BEGIN
        WRITE(OFILE,I:3,'  ');
        WRITE(OFILE,S.NAME[MST.SYMPTR[MST.PTR[I]]]);
        WRITE(OFILE,S.NAME[MST.PNO[MST.PTR[I]]]);
        WRITE(OFILE,'=',MST.VAL[MST.PTR[I]]:3);
        WRITE(OFILE,'   ',MST.F1COV[MST.PTR[I]]:5);
        WRITE(OFILE,',',MST.FOCOV[MST.PTR[I]]:5);
        WRITELN(OFILE);
        END;
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                  ENTERP
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
   (*PMETA*)
  PROCEDURE ENTERP;
  LABEL 1,2,3,4,5,6,7,8,9;
  TYPE NTYPE = PACKED ARRAY[1..11] OF CHAR;
  VAR NAME :ARRAY[1..28] OF NTYPE;
  BUF : ARRAY[0..80] OF CHAR;
  I,J,K1,K,L,M,BLEN : INTEGER;
  G:GPTR;
  PROCEDURE PDOM;
  VAR I,J,K:INTEGER;
  BEGIN
    WRITELN(OFILE,'    NAME    ',' NARG ',' TYPE ',
' COST ',' MIN ',' MAX ', '  STRUCTURE');
    WITH S DO
      FOR I:=1 TO NELT DO
        BEGIN
          WRITE(OFILE,NAME[I],NARG[I]:4,VTYPE
[I]:6,VCOST[I]:6,MVAL[I]:5,EVAL[I]:5);
          IF VTYPE[I]=3 THEN
            FOR J:=1 TO DST.NELE DO
              IF I=DST.PNO[J] THEN
                BEGIN
                  FOR K:=0 TO MNVAL DO
                    IF K IN DST.PREM[J] THEN
                      WRITE(OFILE,K:2);
                  WRITE(OFILE,'=>');
                  FOR K:=0 TO MNVAL DO
                    IF K IN DST.CONS[J] THEN
                      WRITE(OFILE,K:2);
                  WRITE(OFILE,';  ');
                  END;
          WRITELN(OFILE);
          END;
    END;
   (*PDOM*)
  PROCEDURE PRINTPS;
  VAR I,J:INTEGER;
  BEGIN
    WRITELN(OFILE);
    WRITELN(OFILE);
    WRITE(OFILE,'             TRACE=');
    FOR I:=0 TO 10 DO
      IF I IN TRACE THEN
        WRITE(OFILE,I:3);
    WRITELN(OFILE);
    WRITE(OFILE,'             STOPS=');
    FOR I:=0 TO 10 DO
```

```
      IF I IN STP THEN
        WRITE(OFILE,I:3);
    WRITELN(OFILE);
    IF PRULE THEN
      WRITE(OFILE,'           PRINT RULES AND RULE NUMBERS');
    WRITELN(OFILE);
    WRITELN(OFILE);
    WRITELN(OFILE,'             VARIABLE NAME ',
'                VTYPE', '    VCOST');
    FOR I:=1 TO S.NELT DO
      IF (S.VCOST[I]<>0) OR (S.VTYPE[I]<>1) THEN
        BEGIN
          WRITE(OFILE,'                   ');
          FOR J:=1 TO 10 DO
            WRITE(OFILE,S.NAME[I,J]);
          WRITE(OFILE,'          ');
          WRITELN(OFILE,'        ',S.VTYPE[I]:1,S.VCOST[I]:9);
          END;
    WRITELN(OFILE);
    WRITELN(OFILE, '              AQPARMS', '            ',
'                   VLPARMS');
    WRITE(OFILE,'       ','    AQMAXSTAR = ', AQP.MAXSTARAQ:3);
    WRITE(OFILE,'      ');
    IF AQP.LQST THEN
      WRITE(OFILE,' LQST')
      ELSE
      WRITE(OFILE,'     ');
    WRITE(OFILE,'    NCONSIST = ', PRM.NCONSIST:3, ' ALTER = ', PRM.ALTER:3);
    WRITELN(OFILE,  ' VLMAXSTAR = ',PRM.MAXSTAR:3);
    WRITELN(OFILE);
    WRITELN(OFILE, '       AQCRIT        ','    AQTOLERANCE          VLCRIT',
'          VLTOLERANCE');
    FOR I:=1 TO 6 DO
      WRITELN(OFILE, '        ', AQP.CSTF[I]:2, '              ',
       AQP.TOLER[I]:5:2, '                ', PRM.CSTF[I] :2, '               ',
 PRM.TOLER[I]:5:2);
    WRITELN(OFILE);
    WRITELN(OFILE, 'NBR OF CRIT:      AQNF = ', AQP.NF:3, '                    ',
'            VLNF = ', PRM.NF:3);
    WRITE(OFILE,'NEW FNCTNS:       METATRIM = ',MST.METATRIM:3);
    IF PRM.EXTMTY THEN
      WRITE(OFILE,'     EXTMTY');
    IF PRM.EQUIV THEN
      WRITE(OFILE,'     EQUIV');
    WRITELN(OFILE);
    END;
   (*PRINTPS*)
  PROCEDURE GETNUM(VAR J:INTEGER;
  VAR I:INTEGER);
  LABEL 1,2;
  VAR NEG : BOOLEAN;
  BEGIN
    NEG:=FALSE;
    I:=0;
    WHILE J<BLEN DO
      BEGIN
        J:=J+1;
        IF BUF[J]='-' THEN
          NEG:=TRUE;
        IF (BUF[J] IN ['0'..'9'])AND(NOT(BUF[J-1] IN ['0'..'9','X']))THEN
          GOTO 1;
        END;
1:  WHILE (J<=BLEN)AND(BUF[J]IN['0'..'9'])DO
      BEGIN
        I:=I*10+ORD(BUF[J])-ORD('0');
        J:=J+1;
        END;
    IF NEG THEN
      I:=-I;
    END;
   (*GETNUM*)
  BEGIN
    NAME[ 1 ]:='TRACE      ';
    NAME[ 2 ]:='AQCUTF1    ';
    NAME[ 3 ]:='AQMAXSTAR  ';
    NAME[ 4 ]:='AQTOLERANCE';
    NAME[ 5 ]:='AQCRIT     ';
    NAME[ 6 ]:='AQNF       ';
    NAME[ 7 ]:='VCOST      ';
    NAME[ 8 ]:='VLMAXSTAR  ';
    NAME[ 9 ]:='VLTOLERANCE';
    NAME[ 10 ]:='VLCRIT     ';
    NAME[ 11 ]:='VLNF       ';
    NAME[ 12 ]:='NCONSIST   ';
    NAME[ 13 ]:='ALTER      ';
    NAME[ 27 ]:='PRULE      ';
    NAME[ 28 ]:='LOST       ';
    NAME[ 17 ]:='QUIT       ';
    NAME[ 18 ]:='HELP PARAM ';
    NAME[ 19 ]:='PARAMETERS ';
    NAME[ 20 ]:='STP        ';
    NAME[ 21 ]:='           ';
    NAME[ 22 ]:='QUICK      ';
    NAME[ 23 ]:='DETAIL     ';
    NAME[ 24 ]:='EXPLAIN    ';
    NAME[ 25 ]:='BRIEF      ';
    NAME[ 26 ]:='VTYPE      ';
    NAME[ 14 ]:='PRINT      ';
    NAME[ 15 ]:='METATRIM   ';
    NAME[ 16 ]:='           ';
2:  IF INFILE=0 THEN
      WRITELN(OFILE,'ENTER RULE     TO SEE RULE',
' PARA OR PARM', '=VALUE TO SEE OR CNG.PARM',' HELP OR QUIT');
    PUTSEG(OFILE);
    TLINF;
    FOR BLEN:=0 TO 80 DO
      BUF[BLEN]:=' ';
    BLEN:=0;
    WHILE NOT EOS(I) DO
      BEGIN
        BLEN:=BLEN+1;
        GETCHRR(BUF[BLEN]);
        END;
    J:=0;
    GETNUM(J,I);
    GETNUM(J,L);
    IF BUF[1] IN ['0'..'9'] THEN
      BEGIN
        G:=GSET;
        WHILE G<>NIL DO
          IF G.RNO=I THEN
            GOTO 1
            ELSE
            G:=G.NXTN;
        G:=STAR;
        WHILE G<>NIL DO
          IF G.RNO=I THEN
            GOTO 1
            ELSE
            G:=G.NXTN;
        G:=PSTAR;
        WHILE G<>NIL DO
          IF G.RNO=I THEN
            GOTO 1
            ELSE
            G:=G.NXTN;
        G:=COVSET;
        WHILE G<>NIL DO
          IF G.RNO=I THEN
            GOTO 1
            ELSE
            G:=G.NXTN;
        G:=RFSTLIST;
        WHILE G<>NIL DO
          IF G.RNO=I THEN
            GOTO 1
            ELSE
            G:=G.NXTN;
        WRITELN(OFILE,'RULE',I,' NOT FOUND');
        GOTO 2;
1:      PGRAPH(G,S);
        GOTO 2;
        END
      (*IF BUF IN*)
    ELSE
    FOR K:=1 TO 28 DO
      BEGIN
        FOR K1:=1 TO 4 DO
          IF BUF[K1]<>NAME[K,K1] THEN
            GOTO 3;
        GOTO 5;
3:      ;
        END;
    WRITELN(OFILE,'TRY AGAIN');
    GOTO 2;
5:  CASE K OF
1:    IF I<0 THEN
        TRACE:=TRACE-[ABS(I)]
        ELSE
        TRACE:=TRACE+[ABS(I)];
2:    AQP.CUTP1:=I;
3:    AQP.MAXSTAPAQ:=I;
4:    AQP.TOLER[I]:=L/100.0;
5:    IF I>0 THEN
        AQP.CSTF[I]:=L;
6:    AQP.NF:=I;
14:   BEGIN
        IF BUF[7] IN ['M','R','D','F'] THEN
          BEGIN
            CASE BUF[7] OF
'M':          PMETAD;
'R':          G:=RESTLIST;
'D':          PDOM;
'F':          G:=GSET
              END;
                  (*CASE STMT*)
            IF BUF[7] IN ['R','F'] THEN
              WHILE G<>NIL DO
                BEGIN
                  PGRAPH(G,S);
                  G:=G.NXTN;
                  END;
            END
            (*IF*)
        ELSE
        BEGIN
          WRITELN(OFILE,'ENTER PRINT X WHERE X IS');
          WRITELN(OFILE,'      (M) PRINT META DESCRIPTORS');
          WRITELN(OFILE,'      (F) PRINT INPUT DECISION RULES');
          WRITELN(OFILE,'      (D) DOMAIN INFORMATION');
          WRITELN(OFILE,'      (R) RESTRICTIONS');
          END;
        END;
15:   MST.METATRIM:=I;
7,26: BEGIN
        L:=0;
        M:=0;
        FOR J:=1 TO BLEN DO
          IF BUF[J]='(' THEN
            L:=J+1
            ELSE
            IF BUF[J]=')' THEN
              M:=J-1;
        IF M*L = 0 THEN
          BEGIN
            WRITELN(OFILE,'INVALID SYNTAX');
            GOTO 2;
            END;
        FOR J:=1 TO S.NELT DO
          BEGIN
            FOR K1:=L TO M DO
              IF BUF[K1]<>S.NAME[J,K1-L+1] THEN
                GOTO 8;
            GOTO 9;
8:          ;
            END;
        WRITELN(OFILE,'DESCRIPTOR NOT FOUND IN STAB');
        GOTO 2;
9:      IF K=7 THEN
          S.VCOST[J]:=I
          ELSE
          S.VTYPE[J]:=I;
        GOTO 2;
        END;
         (*CASE 7*)
8:    PRM.MAXSTAR:=I;
9:    PRM.TOLER[I]:=L/100.0;
10:   IF I>0 THEN
        PRM.CSTF[I]:=L;
11:   PRM.NF:=I;
12:   PRM.NCONSIST:=I;
13:   PRM.ALTER:=I;
27:   IF BUF[7]<>' ' THEN
        PRULE:=FALSE
        ELSE
        PRULE:=TRUE;
28:   IF BUF[6]<>' ' THEN
        AQP.LOST:=FALSE
        ELSE
        AQP.LOST:=TRUE;
17:   GOTO 4;
18:   BEGIN
        FOR I:=1 TO 28 DO
          BEGIN
            FOR K1:=1 TO 4 DO
              IF BUF[K1+5]<>NAME[I,K1] THEN
                GOTO 6;
            EXPLN(100*I);
            GOTO 7;
6:          ;
            END;
        EXPLN(18);
        WRITELN(OFILE,' THE VALID PARAMETERS ARE:');
        FOR I:=1 TO 28 DO
          WRITELN(OFILE,NAME[I]);
7:      ;
        END;
         (*CASE 18*)
19:   PRINTPS;
20:   IF I<0 THEN
        STP:=STP-[ABS(I)]
        ELSE
        STP:=STP+[ABS(I)];
22:   TRACE:=[];
23:   BEGIN
        TRACE:=[1..10];
        STP:=[];
        END;
24:   BEGIN
        TRACE:=[1..10];
        STP:=[1..10];
        END;
25:   BEGIN
        TRACE:=[3,9,10];
        STP:=[10];
        END
      END;
      (*CASE STMT*)
    GOTO 2;
4:  ;
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                                SOUT;
         OUTPUT SYMBOL TABLE ON STAB
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
   (*ENTERP*)
  PROCEDURE SOUT;
  VAR I,J:INTEGER;
  BEGIN
    STAB:=S;
    PUT(STAB);
    REWRITE(DFILE);
    DFILE:=DST;
    PUT(DFILE);
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                                         ADDCONS(G1,G2:GPTR);
           ADD CONSEQUENCE OF RESTRICTION TO GRAPH IF NOT ALREADY THERE
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
  PROCEDURE ADDCONS(G1,G2:GPTR);
  LABEL 1,99;
  VAR I,J,K,L:INTEGER;
  BEGIN
      (* CONSEQUENCE IS IN GSIZE NODE *)
    FOR J:=1 TO GSIZE DO
      IF (G1.PNO[GSIZE]=G2.PNO[J]) THEN
        IF INSIDE(G1.PNO[GSIZE],G2.VAL[J],G1.VAL[GSIZE],TRUE) THEN
          BEGIN
            I:=1;
            WHILE G2.LNK[J,I]<>0 DO
              BEGIN
                IF G2.ASSGN[G2.LNK[J,I]]<>G1.LNK[GSIZE,I] THEN
                  GOTO 1;
                I:=I+1;
                END;
            G2.VAL[J]:=G1.VAL[GSIZE];
                  (*CONSEQUENCE ALREADY IN G2, RETURN*)
            GOTO 99;
1:          ;
            END;
      (*FOR J*)
      (*CONSEQUENCE NOT IN G2, ADD TO G2*)
    I:=1;
    WHILE G2.LNK[I,1]<>0 DO
      I:=I+1;
    G2.PNO[I]:=G1.PNO[GSIZE];
    G2.VAL[I]:=G1.VAL[GSIZE];
    G2.VBL[I]:=G1.VBL[GSIZE];
    G2.ORDIRP[I]:=G1.ORDIPR[GSIZE];
    J:=1;
      (*ADD SELECTOR TO G2*)
    WHILE G1.LNK[GSIZE,J]<>0 DO
      BEGIN
        G2.LNK[I,J]:=G1.ASSGN[G1.LNK[GSIZE,J]];
        L:=1;
        WHILE G2.LNK[G2.LNK[I,J],L]<>0 DO
          L:=L+1;
        G2.LNK[G2.LNK[I,J],L]:=I;
        J:=J+1;
        END;
99: END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                                ALLC(VAR F1:CPTR;
         TRANSLATE FROM GRAPH STRUCTURE INTO COMPLEX FOR AQ
```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
   (*CONSADD*)
  PROCEDURE ALLC(VAR F1:CPTR;
  GSUB,G:GPTR);
  LABEL 1,2;
  VAR I,J:INTEGER;
  P:CPTR;
  BEGIN
    IF AQP.FREEC = NIL THEN
      BEGIN
        NEW(AQP.FREEC);
        AQP.FREEC.NXTC:=NIL;
        END;
    P:=AQP.FREEC;
    AQP.FREEC:=AQP.FREEC.NXTC;
    P.NXTC:=F1;
    F1:=P;
    FOR J:=1 TO MST.NMST DO
      BEGIN
        F1.CVAL[J]:=G.MSEL.CVAL[MST.PTR[J]];
        AQP.SLOC[J]:=MST.SYMPTR[MST.PTR[J]];
        END;
    J:=MST.NMST;
    FOR I:=1 TO GSIZE DO
      IF (GSUB.COUNT[I]=1) THEN
        BEGIN
          J:=J+1;
          AQP.SLOC[J]:=ABS(GSUB.PNO[I]);
          F1.CVAL[J]:=G.VAL[GSUB.ASSGN[I]];
          END;
    AQP.NVAR:=J;
    P:=F1.NXTC;
    WHILE P<>NIL DO
      BEGIN
        FOR J:=1 TO AQP.NVAR DO
          IF P.CVAL[J]<>F1.CVAL[J] THEN
            GOTO 1;
        P:=F1.NXTC;
        F1.NXTC:=AQP.FREEC;
        AQP.FREEC:=F1;
        F1:=P;
1:      P:=P.NXTC;
        END;
2:  ;
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                            SUBG1(G1,G2:GPTR,ALLSUBG:INTEGER;VAR F:CPTR):BOOLEAN;
         DETERMINE IF G1 IS SUBGRAPH OF G2.  SUBG1 FINDSTHE  FIRST MATCHING
SELECTORS OF G1 AND G2.  SUBG IS CALLED TO MATCH THE REST OF
THE GRAPH.  A BACKTRACK ARRAY LIST IS MAINTAINED TO FACILITATE
BACKING UP IN THE SPANNING TREE IF TWO NODES DONT MATCH.  LIST
CONTAINS THE NODE NUMBER IN G1, AND THE LINK NUMBERS IN G1 AND
G2 FOR EACH MATCH.  AT EACH MATCH, ASSGN IS SET TO SPECIFY THE
1-TO-1 CORRESPONDENCE WHICH EXISTS BETWEEN GRAPHS.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
   (* ALLC *)
  FUNCTION SUBG1(G1,G2:GPTR;
  ALLSUBG:INTEGER;
  VAR F:CPTR;
  INSD:BOOLEAN):BOOLEAN;
  LABEL 99,1,2;
  TYPE LAR = ARRAY[1..600] OF INTEGER;
  VAR L1,L2,PTR:INTEGER;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                                    SUBG(G1,G2:GPTR;
         RECURSIVE ALGORITHM WHICH DETERMINES IF THE TREE WITH
    ROOT ND1 IS A SUBGRAPH OF THE GRAPH WITH ROOT ND2.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
  FUNCTION SUBG(G1,G2:GPTR;
  N1,N2:INTEGER):BOOLEAN;
  LABEL 1,2,3,4;
```

```
    VAR P,P1,I,J,LASTP:INTEGER;
    DONE:BOOLEAN;
    FATHER,L1,L2,ND1,ND2:ARRAY[1..200] OF INTEGER;
    FUNCTION MATCH(P:INTEGER):INTEGER;
    VAR N1,N2,TMATCH,I,J:INTEGER;
    BEGIN
      N1:=G1.LNK[ND1[P],L1[P]];
      N2:=G2.LNK[ND2[P],L2[P]];
      TMATCH:=0;
      IF (G1.ASSGN[N1]=N2)AND(G2.ASSGN[N2]=N1)THEN
        TMATCH:=1
        ELSE
        IF(G1.ASSGN[N1]=0)AND(G2.ASSGN[N2]=0)THEN
          IF G1.PNO[N1]=G2.PNO[N2] THEN
            IF INSIDE(G1.PNO[N1],G2.VAL[N2],G1.VAL[N1],INSD) THEN
              TMATCH:=2;
      IF TMATCH>0 THEN
        IF NOT G1.ORDIRR[N1] THEN
          BEGIN
            I:=1;
            WHILE G1.LNK[N1,I]<>ND1[P] DO
              I:=I+1;
            J:=1;
            WHILE G2.LNK[N2,J]<>ND2[P] DO
              J:=J+1;
            IF I<>J THEN
              TMATCH:=0;
            END;
      IF TMATCH=2 THEN
        BEGIN
          G1.ASSGN[N1]:=N2;
          G2.ASSGN[N2]:=N1;
          END;
      MATCH:=TMATCH;
      END;
     (*MATCH*)
    BEGIN
      G1.ASSGN[N1]:=N2;
      G2.ASSGN[N2]:=N1;
      SUBG:=FALSE;
      P:=1;
      FATHER[1]:=0;
      L1[1]:=1;
      L2[1]:=1;
      ND1[1]:=N1;
      ND2[1]:=N2;
2:    WHILE P<>0 DO
        BEGIN
          WHILE G1.LNK[ND1[P],L1[P]]=0 DO
            BEGIN
              P1:=FATHER[P];
              IF P1=0 THEN
                BEGIN
                  SUBG:=TRUE;
                  GOTO 1;
                  END
              ELSE
              BEGIN
                L1[P]:=L1[P1]+1;
                ND1[P]:=ND1[P1];
                IF G1.ORDIRR[ND1[P]] THEN
                  L2[P]:=1
                  ELSE
                  L2[P]:=L1[P];
                ND2[P]:=ND2[P1];
                FATHER[P]:=FATHER[P1];
                END;
              END;
          REPEAT
            DONE:=TRUE;
            IF P<>0 THEN
            IF G2.LNK[ND2[P],L2[P]]=0 THEN
              BEGIN
                DONE:=FALSE;
                IF L1[P]=1 THEN
                  BEGIN
                    G1.ASSGN[ND1[P]]:=0;
                    G2.ASSGN[ND2[P]]:=0;
                    END;
                P:=P-1;
                IF P<>0 THEN
                  IF NOT G1.ORDIRR[ND1[P]] THEN
                    L2[P]:=MLNK
                    ELSE
                    L2[P]:=L2[P]+1;
                END;
          UNTIL DONE;
        IF P<>0 THEN
          CASE MATCH(P) OF
0:          IF G1.ORDIRR[ND1[P]] THEN
              L2[P]:=L2[P]+1
              ELSE
              L2[P]:=MLNK;
1:          BEGIN
              LASTP:=P;
              P:=P+1;
              FATHER[P]:=FATHER[P-1];
              L1[P]:=L1[P-1]+1;
              ND1[P]:=ND1[P-1];
              IF G1.ORDIRR[ND1[P]] THEN
                L2[P]:=1
                ELSE
                L2[P]:=L1[P];
              ND2[P]:=ND2[P-1];
              END;
2:          BEGIN
              LASTP:=P;
              P:=P+1;
              FATHER[P]:=P-1;
              ND1[P]:=G1.LNK[ND1[P-1],L1[P-1]];
              ND2[P]:=G2.LNK[ND2[P-1],L2[P-1]];
              L1[P]:=1;
              L2[P]:=1;
              END
            END;
            (*CASE STMT*)
        END;
      (*WHILE*)
1:  IF (ALLSUBG<>0) AND (P<>0) THEN
      BEGIN
        P:=LASTP;
        IF G1.ORDIRR[ND1[P]] THEN
          L2[P]:=L2[P]+1
          ELSE
          L2[P]:=MLNK;
        SUBG:=FALSE;
        IF ALLSUBG=1 THEN
          ADDCONS(G1,G2)
          ELSE
          ALLC(F,G1,G2);
        GOTO 2;
        END;
    END;
   (*SUBG*)
   (* THIS PROCEDURE ERCHES FOR STARTING NODES IN G2*
   *)
  BEGIN
    SUBG1:=FALSE;
    IF (G1.MSEL<>NIL)AND(G2.MSEL<>NIL) THEN
      FOR L1:=1 TO MST.NMST DO
        IF NOT (G1.MSEL.CVAL[MST.PTR[L1]]>=G2.MSEL.CVAL[MST.PTR[L1]]) THEN
          GOTO 99;
    FOR L1:=1 TO GSIZE-1 DO
      BEGIN
        G1.ASSGN[L1]:=0;
        G2.ASSGN[L1]:=0;
        END;
      (*PRESCAN TO FIND IF POSSIBLE CORRESPONDENCD*)
    FOR L1:=1 TO GSIZE-1 DO
      IF G1.LNK[L1,1]<>0 THEN
        BEGIN
          FOR L2:=1 TO GSIZE DO
            IF(G2.LNK[L2,1]<>0)AND(G2.ASSGN[L2]=0) THEN
              IF(G2.PNO[L2]=G1.PNO[L1]) THEN
                IF INSIDE(G1.PNO[L1],G2.VAL[L2],G1.VAL[L1],INSD)THEN
                  BEGIN
                    G2.ASSGN[L2]:=1;
                    GOTO 2;
                    END;
          GOTO 99;
2:        ;
          END;
    FOR L2:=1 TO GSIZE DO
      G2.ASSGN[L2]:=0;
    FOR L1:=1 TO GSIZE-1 DO
      IF (G1.LNK[L1,1]<>0)AND(G1.LNK[L1,2]<>0) AND(NOT G1.ORDIRR[L1]) THEN
        GOTO 1;
    FOR L1:=1 TO GSIZE-1 DO
      IF(G1.LNK[L1,1]<>0)AND(NOT G1.ORDIRR[L1])THEN
        GOTO 1;
    FOR L1:=1 TO GSIZE DO
      IF G1.LNK[L1,1]<>0 THEN
        GOTO 1;
1:  FOR L2:=1 TO GSIZE DO
      IF G2.LNK[L2,1]<>0 THEN
        IF G1.PNO[L1]=G2.PNO[L2] THEN
          IF INSIDE(G1.PNO[L1],G2.VAL[L2],G1.VAL[L1],INSD) THEN
            IF  SUBG(G1,G2,L1,L2) THEN
              BEGIN
                SUBG1:=TRUE;
                GOTO 99;
                END
    ELSE
    BEGIN
      FOR PTR:=1 TO GSIZE DO
        BEGIN
          G1.ASSGN[PTR]:=0;
          G2.ASSGN[PTR]:=0;
          END;
      PTR:=-2;
      END;
99: ;
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                                PCPX(F:CPTR);
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
   (*SUBG1*)
  PROCEDURE PCPX(F:CPTR);
  VAR I,J,NSEL:INTEGER;
  BEGIN
    NSEL:=1;
    FOR I:=1 TO AQP.NVAR DO
      IF F.CVAL[I]<>[0..MNVAL] THEN
        BEGIN
          IF I>9 THEN
            WRITE(OFILE,'[X',I:2,'=')
            ELSE
            WRITE(OFILE,'[X',I:1,'=');
          IF F.CVAL[I]=[0..MNVAL] THEN
            WRITE(OFILE,'*')
            ELSE
            FOR J:=S.MVAL[ABS(AQP.SLOC[I])] TO S.EVAL[ABS( AQP.SLOC[I])] DO
              IF J IN F.CVAL[I] THEN
```

```
                WRITE(OFILE,J:3);
          WRITE(OFILE,']');
          NSEL:=NSEL+1;
          IF NSEL>8 THEN
            BEGIN
              WRITELN(OFILE);
              NSEL:=1;
              FOR J:=1 TO 5 DO
                WRITE(OFILE,' ');
              END;
          END;
      (*FOR I:=*)
    WRITELN(OFILE);
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                              AQ(GSUB:GPTR;VL1M:BOOLEAN;F1,F2:CPTR;
         ACTUAL AQ ALGORITHM -- MUCH LIKE AQ7
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
   (*PCPX*)
  FUNCTION AQ(GSUB:GPTR;
  VL1M:BOOLEAN;
  F1,F2:CPTR):CPTR;
  LABEL 1,2,3,4,10,12,13,7,8,21,22,23,99;
  VAR DELTA,I,J,K,L:INTEGER;
  NSTAR,E1,E2,AQT,QSTAR,P,Q,R:CPTR;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                              TRIM(VAR NSTAR:CPTR;
    TRIM NSTAR TO MAXS ELEMENTS
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
  PROCEDURE TRIM(VAR NSTAR:CPTR;
  MAXS:INTEGER);
  LABEL 1,2,99;
  TYPE ATYPE = ARRAY[0..300] OF CPTR;
  VAR CA:ATYPE;
  X:REAL;
  NC,I,J,IB,IC:INTEGER;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                              COSTF(P:CPTR;
         DETERMINE COST OF THIS COMPLEX
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
  FUNCTION COSTF(P:CPTR;
  CT:INTEGER):INTEGER;
  LABEL 6;
  VAR I,J,K:INTEGER;
  INSD:BOOLEAN;
  G1:GPTR;
  CTNEG:BOOLEAN;
  Q:CPTR;
  BEGIN
      (*COSTF*)
    IF CT<0 THEN
      CTNEG:=TRUE
      ELSE
      CTNEG:=FALSE;
    CT:=ABS(CT);
    P.COST:=0;
    CASE CT OF
3:    BEGIN
        G1:=GSET;
        K:=0;
        WHILE G1<>NIL DO
          BEGIN
            IF G1.FP AND (ES IN G1.ESET) THEN
              BEGIN
                J:=1;
                K:=K+1;
                IF GSUB.MSEL<>NIL THEN
                  FOR J:=1 TO MST.NMST DO
                    GSUB.MSEL.CVAL[MST.PTR[J]]:=P.CVAL[J];
                J:=MST.NMST+1;
                FOR I:=1 TO GSIZE DO
```

```
                  IF (GSUB.COUNT[ I ]=1) THEN
                    BEGIN
                      GSUB.VAL[ I ]:=P.CVAL[ J ];
                      J:=J+1;
                      END;
                IF SUBG1(GSUB,G1,0,AQP.FREEC,TRUE) THEN
                  P.COST:=P.COST+1;
                END;
            G1:=G1.NYTN;
            IF(MAYS>1) AND (K>AQP.CUTF1) THEN
              G1:=NIL;
            END;
             (*WHILE G1<>NIL*)
        END;
         (*CASF 1*)
2,4:  BEGIN
        FOR J:=1 TO AQP.NVAR DO
          IF ([0..MNVAL ]-P.CVAL[ J ]) <>[ ] THEN
            IF CT = 2 THEN
              P.COST:=P.COST+1
              ELSE
              P.COST:=P.COST+S.VCOST[ AQP.SLOC[ J ] ];
        END;
         (*CASE 2*)
1,5,6:BEGIN
        CASF CT OF
1:        BEGIN
            Q:=F1;
            INSD:=TRUE;
            END;
5:        BEGIN
            Q:=F1;
            INSD:=TRUE;
            END;
6:        BEGIN
            Q:=F2;
            INSD:=FALSE;
            END
          END;
             (*CASE STMT*)
        WHILE Q<>NIL DO
          BEGIN
            IF ((CT=1) AND Q.FQ)OR(CT IN [5,6 ]) THEN
              FOR I:=1 TO AQP.NVAR DO
                IF INSD THEN
                  IF NOT(Q.CVAL[ I ]<=P.CVAL[ I ]) THEN
                    GOTO 6
                    ELSE
                    ELSE
                    IF NOT(Q.CVAL[ I ]*P.CVAL[ I ]<>[ ]) THEN
                      GOTO 6;
            P.COST:=P.COST+1;
6:          Q:=Q.NXTC;
            END;
        END
         (*CASE 3*)
      END;
      (*CASE STMT*)
    IF CTNEG THEN
      P.COST:=-P.COST;
    COSTF:=P.COST;
    END;
   (*COSTF*)
  BEGIN
      (*TRIM*)
    IC:=1;
    IB:=1;
    P:=NSTAR;
    NC:=0;
    WHILE P<>NIL DO
      BEGIN
```

```
        Q:=P;
        IF P.FP THEN
          BEGIN
            NC:=NC+1;
            CA[NC]:=P;
            P:=P.NXTC;
            END
            (*IF P.FP*)
        ELSE
        BEGIN
          P:=P.NXTC;
          Q.NXTC:=AQP.FREEC;
          AQP.FREEC:=Q;
          END;
        END;
      (*WHILE P<>NIL *)
    CA[NC+1]:=CA[NC];
    CA[0]:=CA[1];
1:  IF NC<=MAXS THEN
      GOTO 99;
    I:=1;
    IF MAXS=0 THEN
      GOTO 2;
    FOR J:=1 TO NC DO
      CA[J].COST:=COSTF(CA[J],AQP.CSTF[IC]);
      (*SORT ARRAY CA *)
    FOR I:=IB TO NC-1 DO
      FOR J:=I+IB TO NC DO
        IF CA[J].COST < CA[I].COST THEN
          BEGIN
            P:=CA[J];
            CA[J]:=CA[I];
            CA[I]:=P;
            END;
    I:=MAXS+1;
    IF AQP.TOLER[IC]=TRUNC(AQP.TOLER[IC]) THEN
      X:=AQP.TOLER[IC]
      ELSE
      X:=AQP.TOLER[IC]*(CA[NC].COST-CA[1].COST);
    IF IC<>AQP.NF THEN
      WHILE (CA[MAXS].COST >= CA[I].COST-X) AND (I<=NC) DO
        I:=I+1;
      (* RETURN ELEMENTS FROM I TO NC*)
2:  FOR J:=I TO NC DO
      BEGIN
        CA[J].NXTC:=AQP.FREEC;
        AQP.FREEC:=CA[J];
        END;
    NC:=I-1;
    IB:=MAXS-1;
    WHILE(CA[MAXS].COST <= CA[IB].COST+X) AND (IB>0) DO
      IB:=IB-1;
    IB:=IB+1;
    IC:=IC+1;
    IF IC<=AQP.NF THEN
      GOTO 1;
99: NSTAR:=NIL;
    FOR I:=1 TO NC DO
      BEGIN
        CA[I].NXTC:=NSTAR;
        NSTAR:=CA[I];
        END;
    END;
   (*TRIM*)
  BEGIN
      (* PLACE ALL EVENTS INTO FQ AND FP SETS *)
    AQ:=NIL;
    IF (F1=NIL) THEN
      GOTO 99;
    WITH AQP DO
      BEGIN
```

```
          AQT:=NIL;
          P:=F1;
          WHILE P<>NIL DO
            BEGIN
              P.FP:=TRUE;
              P.FQ:=TRUF;
              P:=P.NXTC;
              END;
              (* ALLCOATE START OF OSTAR *)
          DELTA:=1;
1:        NSTAR:=NIL;
          IF AQP.FREEC=NIL THEN
            BEGIN
              NEW(AQP.FREEC);
              AQP.FREEC.NXTC:=NIL;
              END;
          OSTAR:=AQP.FREEC;
          AQP.FREEC:=AQP.FREEC.NXTC;
          OSTAR.NXTC:=NIL;
          OSTAR.FP:=TRUE;
          FOR I:=1 TO AQP.NVAR DO
            OSTAR.CVAL[I]:=[0..MNVAL];
              (* FIND UNCOVERED EVENT *)
          E1:=F1;
13:       IF NOT (((DELTA=1) AND (E1.FP))OR((DELTA=2) AND (E1.FQ))) THEN
            BEGIN
              E1:=E1.NXTC;
              IF NOT V11M THEN
                GOTO 12;
              IF E1=NIL THEN
                GOTO 12
                ELSE
                GOTO 13;
              END;
          E2:=F2;
          WHILE E2<>NIL DO
            BEGIN
                    (* SEE IF E2 IS IN OSTAR *)
              P:=OSTAR;
              WHILE P<>NIL DO
                BEGIN
                  FOR I:=1 TO AQP.NVAR DO
                    IF (E2.CVAL[I]*P.CVAL[I]) =[ ] THEN
                      GOTO 2;
                  GOTO 3;
2:                P:=P.NXTC;
                  END;
                    (* WHILE P<>NIL*)
              GOTO 10;
                    (* E2 IS IN OSTAR, FINE ELEMENTARY STAR OF E1 AGAI
                     NST E2 *)
                    (* MULTIPLY BY OSTAR *)
3:            FOR I:=1 TO AQP.NVAR DO
                IF E1.CVAL[I]<= ([0..MNVAL]- E2.CVAL[I]) THEN
                  BEGIN
                    P:=OSTAR;
                             (* PUT CPX FROM OSTAR INTO NSTAR, MPY BF E2 COMPL
                             *)
                    WHILE P<>NIL DO
                      BEGIN
                        IF AQP.FREEC=NIL THEN
                          BEGIN
                            NEW(AQP.FREEC);
                            AQP.FREEC.NXTC:=NIL;
                            END;
                        R:=AQP.FREEC;
                        AQP.FREEC:=R.NXTC;
                        R.NXTC:=NSTAR;
                        NSTAR:=R;
                        FOR J:=1 TO AQP.NVAR DO
                          R.CVAL[J]:=P.CVAL[J];
                      FIXIT:=R.CVAL[I];
                      EXTND (AQP.SLOC[I],E1.CVAL[I], E2.CVAL[I]);
                      R.CVAL[I]:=FIXIT;
                      P:=P.NXTC;
                      END;
                           (* WHILE P<>NIL *)
                  END;
                  (* FOR I *)
                  (* NOW APPLY ABSOURPTION LAWS TO NSTAR *)
            P:=NSTAR;
            WHILE P<>NIL DO
              BEGIN
                P.FP:=TRUE;
                P:=P.NXTC;
                END;
            P:=NSTAR;
            WHILE P<> NIL DO
              BEGIN
                IF P.FP THEN
                  BEGIN
                    Q:=NSTAR;
                    WHILE Q<>NIL DO
                      BEGIN
                        IF Q.FP AND (Q<>P) THEN
                          BEGIN
                            FOR I:=1 TO AQP.NVAR DO
                              IF NOT(Q.CVAL[I]<=R.CVAL[I]) THEN
                                 GOTO 4;
                            Q.FP:=FALSE;
                            END;
                                    (*IF Q.FP*)
4:                      Q:=Q.NXTC;
                        END;
                              (*WHILE Q<>NIL*)
                    END;
                        (* IF P.FP *)
                P:=P.NXTC;
                END;
                  (*WHILE P*)
                  (* ABSOURPTION COMPLETE *)
                  (* TRIM NUMBER OF COMPLEXES *)
            TRIM(NSTAR,AQP.MAXSTARAQ);
                  (*RETURN OLIST TO AQP.FREEC *)
            IF NSTAR=NIL THEN
              GOTO 10;
            P:=OSTAR;
            WHILE P.NXTC<>NIL DO
              P:=P.NXTC;
            P.NXTC:=AQP.FREEC;
            AQP.FREEC:=OSTAR;
            OSTAR:=NSTAR;
            NSTAR:=NIL;
10:         E2:=E2.NXTC;
            END;
            (* WHILE E2<>NIL *)
            (* UPDATE FP AND FQ SETS *)
        P:=OSTAR;
        WHILE P<>NIL DO
          BEGIN
            Q:=F1;
            WHILE Q<>NIL DO
              BEGIN
                IF Q.FP THEN
                  FOR I:=1 TO AQP.NVAR DO
                    IF NOT (Q.CVAL[I]<=P.CVAL[I]) THEN
                      GOTO 7;
                Q.FP:=FALSE;
7:              Q:=Q.NXTC;
                END;
                  (* WHILE Q<>NIL*)
            P:=P.NXTC;
            END;
            (* WHILE P<>NIL *)
            (* FINE NEXT F1 TO COVER *)
        IF OSTAR=NIL THEN
          BEGIN
            F1.FP:=FALSE;
            F1.FQ:=FALSE;
            GOTO 1;
            END;
        TRIM(OSTAR,1);
        P:=OSTAR;
            (*LOST*)
        IF AQP.FREEC=NIL THEN
          BEGIN
            NEW(AQP.FREEC);
            AQP.FREEC.NXTC:=NIL;
            END;
        FOR I:=1 TO AQP.NVAR DO
          IF (F2=NIL)OR(P.CVAL[I]<>[0..MNVAL]) THEN
            AQP.FREEC.CVAL[I]:=[ ]
            ELSE
            AQP.FREEC.CVAL[I]:=[0..MNVAL];
        Q:=F1;
        WHILE Q<>NIL DO
          BEGIN
            FOR I:=1 TO AQP.NVAR DO
              IF NOT (Q.CVAL[I]<=P.CVAL[I]) THEN
                GOTO 8;
            Q.FQ:=FALSE;
            FOR I:=1 TO AQP.NVAR DO
              AQP.FREEC.CVAL[I]:=AQP.FREEC.CVAL[I]+Q.CVAL[I];
8:          Q:=Q.NXTC;
            END;
            (* WHILE Q<>NIL*)
        OSTAR.NXTC:=AQT;
        AQT:=OSTAR;
        IF AQP.LOST THEN
          BEGIN
            FOR I:=1 TO AQP.NVAR DO
              CASE S.VTYPE[AQP.SLOC[I]] OF
1:              ;
2:              BEGIN
                  FOR J:=0 TO MNVAL DO
                    IF J IN AQP.FREEC.CVAL[I] THEN
                      GOTO 21;
21:               FOR K:=MNVAL DOWNTO 0 DO
                    IF K IN AQP.FREEC.CVAL[I] THEN
                      GOTO 22;
22:               FOR L:=J TO K DO
                    AQP.FREEC.CVAL[I]:=AQP.FREEC.CVAL[I] + [L];
                  END;
3:              BEGIN
                  IF F2<>NIL THEN
                    AQP.FREEC.CVAL[I]:=OSTAR.CVAL[I]
                    ELSE
                    FOR J:=1 TO DST.NELE DO
                      IF DST.PNO[J]=AQP.SLOC[I] THEN
                        IF AQP.FREEC.CVAL[I]<=DST.PREM[J] THEN
                          BEGIN
                            AQP.FREEC.CVAL[I]:=AQP.FREEC.CVAL[I]+DST.CONS[J
                            GOTO 23;
                            END;
                  END
                END;
                  (*CASE STMT*)
23:         FOR I:=1 TO AQP.NVAR DO
              OSTAR.CVAL[I]:=AQP.FREEC.CVAL[I];
            END;
            (*LOST*)
        GOTO 1;
            (* PASS 2 *)
```

```
12:     IF DELTA = 1 THEN
          BEGIN
            DELTA:=2;
            GOTO 1;
            END;
            (* FIND BEST COMPLEX IN COVER *)
        P:=AQT;
        WHILE P<> NIL DO
          BEGIN
            P.FP:=TRUE;
            P:=P.NXTC;
            END;
        IF NOT VI1M THEN
          TRIM(AQT,1);
        AQ:=AQT;
        END;
      (*WITH AQP*)
99: END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                              AQSET(GSFT:GPTR;
       SETS UP STRUCTURES FOR AQ PROCEDURE.  CVAL CONTAINS BIT POS
   REPRESENTATION OF REFERENCES IN GSUB.  SLOC CONTAINS
   POINTER TO SYMBOL TABLE LOCATION OF ASSOCIATED DESCRIPTOR
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
    (* AQ PROCEDURE *)
  PROCEDURE AQSET(GSET:GPTR;
  ES:INTEGER;
  GSUB:GPTR);
  LABEL 1,3,4,99;
  VAR G,G1:GPTR;
  P,P1,P2:CPTR;
  I,J,K,L:INTEGER;
  DONE:BOOLEAN;
  BEGIN
      (* SET UP CLASS BEING COVERED *)
    G:=GSET;
    WITH GSUB DO
      FOR I:=1 TO GSIZE DO
        IF OPDIFF[I] AND (NOT VBL[I]) THEN
          BEGIN
            J:=1;
            WHILE LNK[I,J]<>0 DO
              BEGIN
                IF LNK[LNK[I,J],2]=0 THEN
                  BEGIN
                    LNK[LNK[I,J],1]:=0;
                    COUNT[LNK[I,J]]:=0;
                    LNK[I,J]:=GSIZE;
                    END;
                J:=J+1;
                END;
            J:=1;
            K:=1;
            WHILE LNK[I,J]<>0 DO
              BEGIN
                IF LNK[I,J]<>GSIZE THEN
                  BEGIN
                    LNK[I,K]:=LNK[I,J];
                    K:=K+1;
                    END;
                J:=J+1;
                END;
            LNK[I,K]:=0;
            END;
    P1:=NIL;
    P2:=NIL;
    P:=NIL;
    WHILE G<>NIL DO
      BEGIN
        IF (ES IN G.ESET)AND(G.FP) THEN
          BEGIN
```

```
            IF SUBG1(GSUB,G,2,F,TRUE)  THEN
              ;GOTO 4;
            END;
        G:=G.NXTN;
        END;
      (* WHILE *)
      (* CLEAR ALL VAL FIELDS OF GSUP *)
4:  FOR I:=1 TO GSIZE DO
      GSUB.VAL[I]:=[0..MNVAL];
    NEW(GSUB.MSEL);
    IF GSUB.MSEL<>NIL THEN
      FOR I:=1 TO GSIZE DO
        GSUB.MSEL.CVAL[I]:=[0..MNVAL];
    G:=G.NXTN;
    WHILE G<>NIL DO
      BEGIN
        IF (ES IN G.ESET) AND G.FP THEN
          IF SUBG1(GSUB,G,2,F1,TPUE) THEN
            ; G:=G.NXTN;
        END;
    F.NXTC:=F1;
    F1:=F;
    G:=GSET;
    WHILE G<>NIL DO
      BEGIN
        IF NOT(ES IN G.ESET) THEN
          IF SUBG1(GSUB,G,2,F2,FALSE)  THEN
            ; G:=G.NYTN;
        END;
      (*WHILE*)
    IF 4 IN TRACE THEN
      BEGIN
        EXPLN(4);
        WRITELN(OFILE,'THE C-FORMULA STRUCTURE IS:');
        PGRAPH(GSUB,S);
        WRITELN(OFILE,'THERE ARE ',AOP.NVAR:3,' VL1 TYPE VARIABLES X1,',
 'X2,...,X', AOP.NVAR:2);
        WRITELN(OFILE,'VARIABLES ARE ASSOCIATED WITH NODES IN THE C-FORMULA',
 ' AS FOLLOWS:');
        WRITELN(OFILE);
        WRITELN(OFILE,'               NODE','                 VARIABLE');
        J:=MST.NMST+1;
        FOR I:=1 TO GSIZE DO
          IF GSUB.COUNT[I]=1 THEN
            BEGIN
              WRITE(OFILE,'              ');
              K:=1;
              WHILE S.NAME[ABS(AOP.SLOC[J]),K]<>' ' DO
                BEGIN
                  WRITE(OFILE,S.NAME[ABS(AOP.SLOC[J]),K]);
                  K:=K+1;
                  END;
              IF GSUF.VBL[I] THEN
                BEGIN
                  IF  GSUB.DUMNUM[I]>9 THEN
                    WRITE(OFILE,GSUB.DUMNUM[I]:2)
                    ELSE
                    WRITE(OFILE,GSUB.DUMNUM[I]:1);
                  K:=K+1;
                  END;
              FOR L:=K TO 20 DO
                WRITE(OFILE,' ');
              IF J>9 THEN
                WRITELN(OFILE,'X',J:2)
                ELSE
                WRITELN(OFILE,'X',J:1);
              J:=J+1;
              END;
        WRITELN(OFILE,'AQ IS APPLIED TO THE FOLLOWING INPUT CPXS/EVENTS');
        WRITELN(OFILE,'                    ','              SET 1');
        F:=F1;
        WHILE F<>NIL DO
          BEGIN
            PCPX(F);
            F:=F.NXTC;
            END;
        WRITELN(OFILE,'                    ','                    SET 2');
        F:=F2;
        WHILE F<>NIL DO
          BEGIN
            PCPX(F);
            F:=F.NXTC;
            END;
        END;
      (*TRACE OF 4*)
3:    F:=AQ(GSUB,FALSE,F1,F2);
    IF F=NIL THEN
      GOTO 99;
    IF 5 IN TRACE THEN
      BEGIN
        TXPLN(5);
        WRITELN(OFILE,'THE RESULTING COMPLEX FROM THIS PASS IS:');
        PCPX(F);
        END;
      (*TRACE 5*)
    ; (* TRANSLATE COVER INTO GRAPH *)   J:=0;
    FOR J:=1 TO MST.NMST DO
      GSUB.MSEL.CVAL[MST.PTR[J]]:=F.CVAL[J];
    J:=MST.NMST;
    FOR I:=1 TO GSIZE DO
      IF GSUB.COUNT[I]=1 THEN
        BEGIN
          J:=J+1;
          GSUB.VAL[I]:=F.CVAL[J];
          END;
    F.NXTC:=AOP.FREEC;
    AOP.FREEC:=F;
99: F:=F1;
    WHILE F.NXTC<>NIL DO
      F:=F.NXTC;
    F.NXTC:=AOP.FREEC;
    AOP.FREEC:=F1;
    IF F2<>NIL THEN
      BEGIN
        F:=F2;
        WHILE F.NXTC<>NIL DO
          F:=F.NXTC;
        F.NXTC:=AOP.FREEC;
        AOP.FREEC:=F2;
        END;
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                               ENTERD
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
   (*ACSET*)
  PROCEDURE ENTERD;
  VAR I:INTEGER;
  BEGIN
    NEWG(G);
    VLINT(G,ERR,FS);
    WITH DST DO
      BEGIN
        NELE:=NELE+1;
        IF NELE>NDES THEN
          WRITELN(OFILE,'DOMAIN STRUC TABLE OVFL');
        PREM[NELE]:=G.VAL[1];
        CONS[NELE]:=G.VAL[2];
        PNO[NELE]:=G.PNO[1];
        FOR I:=1 TO NELE DO
          IF PNO[I]=PNO[NELE] THEN
            IF CONS[I]<=PREM[NELE] THEN
              PREM[NELE]:=PREM[NELE]+PREM[I];
            END;
          (*WITH*)
        G.NXTN:=FREEG;
        FREEG:=G;
        END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                                  VL1
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
     (*ENTERD*)
    PROCEDURE VL1;
    LABEL 1,2,3;
    VAR F1,F2,F,P,Q,AQE:CPTR;
    I,J,K,ES:INTEGER;
    AGNST:VALTP;
    BEGIN
      AQE:=NIL;
      RESET(VL1EVE);
      F:=NIL;
      WITH S DO
        WITH AQP DO
          BEGIN
                 (*SETUP NELT, NAME,PNO*)
            WRITELN(OFILE,'HOW MANY VARIABLES');
            PUTSEG(OFILE);
            ; GETSEG(IFILE);
            WHILE EOLN(IFILE) DO
              GETSEG(IFILE);
            READ(IFILE,NVAR);
            S.NELT:=AQP.NVAR;
            FOR I:=1 TO NVAR DO
              BEGIN
                NAME[I]:='              ';
                VTYPE[I]:=1;
                S.NAME[I,1]:='X';
                IF I>9 THEN
                  BEGIN
                    S.NAME[I,2]:=CHR(TRUNC(I/10)+ORD('0'));
                    S.NAME[I,2]:=CHR(I-TRUNC(I/10)*10+ORD('0'));
                    END
                ELSE
                S.NAME[I,2]:=CHR(I+ORD('0'));
                PNO[I]:=I;
                SLOC[I]:=I;
                DPNO[I]:=I;
                END;
            WHILE NOT EOF(VL1EVE) DO
              BEGIN
                NEW(Q);
                Q.NXTC:=AQE;
                AQE:=Q;
                READ(VL1EVE,I);
                IF I>=0 THEN
                  Q.CVAL[NVAR+1]:=[I]
                  ELSE
                  Q.CVAL[NVAR+1]:=[0..MNVAL];
                FOR I:=1 TO NVAR DO
                  BEGIN
                    READ(VL1EVE,J);
                    IF J IN [0..MNVAL] THEN
                      Q.CVAL[I]:=[J]
                      ELSE
                      Q.CVAL[I]:=[0..MNVAL];
                    IF J<MVAL[I] THEN
                      MVAL[I]:=J;
                    IF J>EVAL[I] THEN
                      EVAL[I]:=J;
                    IF J>NVAL[I] THEN
                      NVAL[I]:=J;
                    END;
                READLN(VL1EVE);
                END;
          END;
      (*READ EVENTS*)
2:  WRITELN(OFILE,'ENTER P TO CHANGE PARAMETERS');
    WRITELN(OFILE,'      C TO COVER EVENTS');
    WRITELN(OFILE,'      E TO ENTER DOMAIN STRUCTURE');
    WRITELN(OFILE,'      Q TO RETURN TO MAIN LEVEL');
    PUTSEG(OFILE);
    ILINE;
    GETCHRR(CHRR);
    IF CHRR IN ['C','Q','E','P'] THEN
      WITH AQP DO
        WITH S DO
          CASE CHRR OF
'P':        ENTERP;
'E':        ENTERD;
'C':        BEGIN
              WRITELN(OFILE,'ENTER DECISION NUMBER OF SET TO BE COVERED');
              PUTSEG(OFILE);
              GETSEG(IFILE);
              READ(IFILE,ES);
              WRITELN(OFILE,'AGAINST WHICH SETS, ENTER NUMBERS',
 'FOR THESE SETS OR ENTER -1 TO COVER AGAINST ALL');
              PUTSEG(OFILE);
              GETSEG(IFILE);
              AGNST:=[];
              WHILE NOT EOLN(IFILE) DO
                BEGIN
                  READ(IFILE,I);
                  IF I=-1 THEN
                    BEGIN
                      AGNST:=[0..MNVAL]-[ES];
                      GOTO 3;
                      END;
                  AGNST:=AGNST+[I];
                  END;
3:            F1:=NIL;
              F2:=NIL;
              Q:=AQE;
              AQE:=NIL;
              WHILE Q<>NIL DO
                BEGIN
                  P:=Q.NXTC;
                  IF ES IN Q.CVAL[NVAR+1] THEN
                    BEGIN
                      Q.NXTC:=F1;
                      F1:=Q;
                      END
                  ELSE
                  IF Q.CVAL[NVAR+1] <= AGNST THEN
                    BEGIN
                      Q.NXTC:=F2;
                      F2:=Q;
                      END
                  ELSE
                  BEGIN
                    Q.NXTC:=AQE;
                    AQE:=Q;
                    END;
                  Q:=P;
                  END;
              IF (F1<>NIL)  THEN
                BEGIN
                  F:=AQ(G,TRUE,F1,F2);
                  WRITELN(OFILE,'OUTPUT COMPLEXES FOR SET',ES:3);
                  Q:=F;
                  WHILE Q<>NIL DO
                    BEGIN
                      P:=Q;
                      PCPX(Q);
                      Q:=Q.NXTC;
                      END;
                  P.NXTC:=FREEC;
                  FREEC:=F;
                  END;
              IF F1<>NIL THEN
                BEGIN
                  P:=F1;
                  WHILE P.NXTC<>NIL DO
                    P:=P.NXTC;
                  P.NXTC:=AQE;
                  AQE:=F1;
                  END;
              IF F2 <> NIL THEN
                BEGIN
                  P:=F2;
                  WHILE P.NXTC<>NIL DO
                    P:=P.NXTC;
                  P.NXTC:=AQE;
                  AQE:=F2;
                  END;
              END;
                  (*CASE C*)
'Q':        GOTO 1
            END;
      (*CASE STMT*)
    GOTO 2;
1:  F:=AQE;
    WHILE F.NXTC<>NIL DO
      F:=F.NXTC;
    F.NXTC:=AQP.FREEC;
    AQP.FREEC:=AQE;
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                                NEWGP(GN,G1:GPTR;
         FIND A NEW GRAPH WITH MNODE SELECTORS IN IT.
    COUNT IN G1 RECORDS THE NUMBER OF TIMES WHICH A SELECTOR HAS
    BEEN USED IN PREVIOUS GRAPHS.  COUNT IN GN INDICATES THE NUMBER
    OF OCCURRENCES OF THIS VBL IN THE NEW GP
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
   (*VL1*)
  PROCEDURE NEWGP(ALTER:INTEGER;
  G0,G1:GPTR;
  VAR SLST:GPTR);
  LABEL 1,2;
  VAR I,J,K,L,M,CPTR:INTEGER;
  CANDID:ARRAY[1..GSIZE] OF INTEGER;
  G:GPTR:
  BEGIN
      (*NEWGP*)
      (*GENERATE A LIST OF ALL SELECTORS WHICH MAY BE CO
       NNECTED TO THE GRAPH. G0 IS OLD GRAPH,     G1 IS E
       V ENT WHICH IS BEING COVERED COUNT=1,NDE FROM OLD
       GR APH COUNT=2 NODE IS VARIABLE CONNECTED TO OLD
      G RAP H COUNT=3 NODE IS NEW SELECTOR *)
    FOR I:=1 TO GSIZE DO
      IF G0.COUNT[I]<>0 THEN
        IF G0.PNO[I]<0 THEN
          BEGIN
            J:=1;
            WHILE G1.LNK[I,J]<>0 DO
              BEGIN
                IF G0.COUNT[G1.LNK[I,J]]=0 THEN
                  G0.COUNT[G1.LNK[I,J]]:=2;
                J:=J+1;
                END;
            END;
    CPTR:=0;
    FOR I:=1 TO GSIZE DO
      IF G0.VBL[I] AND (G0.COUNT[I]>0) THEN
        BEGIN
          J:=1;
          WHILE G1.LNK[I,J]<>0 DO
```

```
            BEGIN
              IF G0.COUNT[G1.LNK[I,J]]=0 THEN
                G0.COUNT[G1.LNK[I,J]]:=3;
              J:=J+1;
              END;
          END;
    FOR I:=1 TO GSIZE DO
      BEGIN
        IF (G0.COUNT[I]=3) THEN
          BEGIN
            CPTR:=CPTR+1;
            CANDID[CPTR]:=I;
            END;
        IF G0.COUNT[I]<>1 THEN
          G0.COUNT[I]:=0;
        END;
      (*SORT CANDID ARRAY IF ALTER < CPTR*)
    IF (ALTER<>0) AND (ALTER<CPTR) THEN
      FOR I:=1 TO CPTR-1 DO
        FOR J:=I+1 TO CPTR DO
          IF (S.VCOST[G0.PNO[CANDID[I]]]> S.VCOST
          [ G0.PNO[CANDID[J]]]) OR (S.VCOST[G0.PNO
          [CANDID[I]]]=S.VCOST[G0.PNO[CANDID
[J]]]) AND (S.NARG[G0.PNO[CANDID[I]]]>S.NARG[G0.PNO[CANDID[J]]]) THEN
            BEGIN
              L:=CANDID[I];
              CANDID[I]:=CANDID[J];
              CANDID[J]:=L;
              END;
      (*FORM NEW GRAPH FOR EACH ALTERNATIVE SELECTOR*)
    M:=0;
    FOR I:=1 TO CPTR DO
      BEGIN
        NEWG(G);
        G:=G0;
        G.COUNT[CANDID[I]]:=1;
        G.RNO:=CRULENO-1;
        J:=1;
        IF G.PNO[CANDID[I]]>0 THEN
          WHILE G1.LNK[CANDID[I],J]<>0 DO
            BEGIN
              G.COUNT[G1.LNK[CANDID[I],J]]:=1;
              J:=J+1;
              END;
        FOR J:=1 TO GSIZE DO
          IF (G1.LNK[J,1]<>0) AND (G.COUNT[J]<>0) THEN
            BEGIN
              K:=1;
              L:=1;
              WHILE G1.LNK[J,K]<>0 DO
                BEGIN
                  IF G.COUNT[G1.LNK[J,K]]=1 THEN
                    BEGIN
                      G.LNK[J,L]:=G1.LNK[J,K];
                      L:=L+1;
                      END;
                  K:=K+1;
                  END;
                     (*IF G1*)
              G.LNK[J,L]:=0;
              IF (G.PNO[J]<0) AND (L=2) THEN
                BEGIN
                  G.NXTN:=FREEG;
                  FREEG:=G;
                  GOTO 1;
                  END;
              END;
            (*FOR J*)
        G.NXTN:=SLST;
        SLST:=G;
        M:=M+1;
```

```
          IF  (ALTER<>0)AND(M>=ALTER)  THEN
            GOTO 2;
1:          ;
          END;
        (*FOR I*)
2:   END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                                         PGRAPH;
     PRINTS A VL2 FORMULA
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
    (*NEWGP*)
   PROCEDURE PGRAPH;
   LABEL 1;
   VAR I,J,K,L,M,NSEL:INTEGER;
   BEGIN
     J:=0;
     WRITE(OFILE,'RULE     ',G.RNO:5);
     IF G.ESET<>[ ] THEN
       WRITE(OFILE,' EVENT SETS:');
     FOR I:=0 TO MNVAL DO
       IF I IN G.ESET THEN
         WRITE(OFILE,I:3);
     WRITE(OFILE,' COSTS (');
     FOR I:=1 TO PRM.NF DO
       WRITE(OFILE,PRM.CSTF[I]:2);
     WRITE(OFILE,')');
     FOR I:=1 TO PRM.NF DO
       IF G.COST[ABS(PRM.CSTF[I])]<>-100 THEN
         WRITE(OFILE,G.COST[ABS(PRM.CSTF[I])]:5)
         ELSE
         WRITE(OFILE,0:5);
     WRITELN(OFILE);
     NSEL:=0;
     IF PRULE  THEN
       WITH G DO
         BEGIN
           FOR I:=1 TO GSIZE DO
             IF VBL[I] AND (G.LNK[I,1]<>0) THEN
               BEGIN
                 J:=J+1;
                 DUMNUM[I]:=J;
                 END;
           FOR I:=1 TO GSIZE DO
             IF (LNK[I,1]<>0) THEN
               IF (NOT VBL[I])OR VBL[I] AND(VAL[I]<>[0..MNVAL])THEN
                 BEGIN
                   NSEL:=NSEL+1;
                   WRITE(OFILE,'[');
                   L:=ABS(PNO[I]);
                   ; FOR J:=1 TO 10 DO IF S.NAME
[L , J]<>' ' THEN WRITE(OFILE,S.NAME[L, J]);
                   IF NOT VBL[I] THEN
                     BEGIN
                       WRITE(OFILE,'(');
                       J:=1;
                       WHILE LNK[I,J]<>0 DO
                         BEGIN
                           M:=LNK[I,J];
                           FOR K:=1 TO 10 DO
                             IF S.NAME[PNO[M],K]<>' ' THEN
                               WRITE(OFILE,S.NAME[PNO[M],K]);
                           IF DUMNUM[M]>9 THEN
                             WRITE(OFILE,DUMNUM[M]:2)
                             ELSE
                             WRITE(OFILE,DUMNUM[M]:1);
                           J:=J+1;
                           IF LNK[I,J]<>0 THEN
                             IF PNO[I]>0 THEN
                               WRITE(OFILE,',')
                               ELSE
                               WRITE(OFILE,'.')
```

```
                              ELSE
                              IF PNO[I]<0 THEN
                                WRITE(OFILE,')=')
                                ELSE
                                IF S.MVAL[PNO[I]]=S.NVAL[PNO[I]] THEN
                                  WRITE(OFILE,')')
                                  ELSE
                                  WRITE(OFILE,')=');
                          END;
                                 (*WHILE*)
                      END
                           (*NOT VBL*)
                  ELSE
                  IF DUMNUM[I]>9 THEN
                    WRITE(OFILE,DUMNUM[I]:2)
                    ELSE
                    WRITE(OFILE,DUMNUM[I]:1);
                  IF PNO[I]>0 THEN
                    IF S.NVAL[PNO[I]]<>S.MVAL[PNO[I]] THEN
                      IF VAL[I]=[0..MNVAL] THEN
                        WRITE(OFILE,' * ')
                        ELSE
                        BEGIN
                          IF S.VTYPE[PNO[I]]=3 THEN
                            FOR M:=S.EVAL[PNO[I]] +1 DOWNTO S.NVAL[PNO[I]] DO
                              IF M IN VAL[I] THEN
                                BEGIN
                                  WRITE(OFILE,M:2);
                                  GOTO 1;
                                  END;
                          FOR M:=S.MVAL[PNO[I]] TO S.NVAL[PNO[I]] DO
                            IF M IN VAL[I] THEN
                              WRITE(OFILE,M:2);
1:                        ;
                          END;
                  IF PNO[I]<0 THEN
                    WRITE(OFILE,'SAME');
                  WRITE(OFILE,']');
                  IF NSEL>=4 THEN
                    BEGIN
                      NSEL:=0;
                      WRITELN(OFILE);
                      PUTSEG(OFILE);
                      WRITE(OFILE,'     ');
                      END;
                  END;
               (*LNK<>^*)
          END;
      (*WITH*)
    WRITELN(OFILE);
    IF G.MSEL<>NIL THEN
      FOR I:=1 TO MST.NMST DO
        IF G.MSEL.CVAL[MST.PTR[I]]<>[0..MNVAL] THEN
          BEGIN
            IF I>9 THEN
              WRITE(OFILE,'[MS',I:2,'=')
              ELSE
              WRITE(OFILE,'[MS',I:1,'=');
            FOR J:=S.MVAL[MST.SYMPTR[MST.PTR[I
]]] TO S.NVAL[MST.SYMPTR[MST.PTR[I]]] DO
              IF J IN G.MSEL.CVAL[MST.PTR[I]] THEN
                WRITE(OFILE,J:2);
            WRITE(OFILE,']');
            END;
    WRITELN(OFILE);
    PUTSEG(OFILE);
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                               TOKEN(
    FINDS THE NEXT TOKEN IN THE INPUT STREAM
```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
   (*PGRAPH*)
  PROCEDURE TOKEN( VAR CURS:INTEGER;
  VAR CTYPE:INTEGER;
  VAR SROW:INTEGER;
  VAR ERR: INTEGER;
  VAR BUF:CARRAY);
  LABEL 1,2;
  CONST DELIMTP = 0;
  DESCTP = 1;
  DUMMYTP = 2;
  DIGITTP = 3;
  TYPE ALPHA = SET OF 'A'..'Z';
  DIGIT = SET OF '0'..'9';
  VAR TRACE,I,L,J,FCURS,LCURS:INTEGER;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                               FINDROW(B,E:INTEGER;
              FIND A ROW IN THE SYMBOL TABLE WITH NAME IN BUF[B]..BUF[E]
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
  PROCEDURE FINDROW(B,E:INTEGER;
  VAR TMP:INTEGER);
  LABEL 1,2;
  VAR J,I:INTEGER;
   (* FIND ROW IN SYMTAB WHICH MATCHES BUF, PUT IN I*
     *)
  BEGIN
    IF TRACE>2 THEN
      WRITELN(OUTPUT,'ENTERING FINDROW',B,E);
    FOR I:=1 TO S.NELT DO
      BEGIN
        FOR J:=1 TO E-B+1 DO
          IF S.NAME[I,J]<>BUF[B-1+J] THEN
            GOTO 1;
        GOTO 2;
1:      ;
        END;
      (* FOR I *)
    I:=0;
2:  TMP:=I;
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                               FIXSYM(I,J:INTEGER);
              ADD A NEW ROW TO SYMBOL TABLE
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
   (* FINDROW *)
  PROCEDURE FIXSYM(I,J:INTEGER);
  VAR K,L : INTEGER;
  C:CHAR;
  BEGIN
    IF TRACE>2 THEN
      WRITELN(OUTPUT,'ENTERING FIXSYM',I,J);
      (* ADD ROW TO STAB OR ELSE REPLACE DESC IN BUF *)
    S.NELT:=S.NELT+1;
    IF S.NELT>SYMSZE THEN
      WRITE(OFILE,'SYMBOL TABLE OVERFLOW, ');
    FOR K:=I TO J DO
      S.NAME[S.NELT,K-I+1]:=BUF[K];
    S.PNO[S.NELT]:=S.NELT;
    CURS:=I;
    IF TRACE>2 THEN
      WRITELN(OUTPUT,'LEAVING FIXSYM');
    END;
   (* FIXSYM *)
  BEGIN
      (*TOKEN*)
    TRACE:=2;
    IF TRACE>2 THEN
      WRITELN(OUTPUT,'ENTERING TOKEN',CURS,CTYPE,SROW,ERR);
1:  IF BUF[CURS] = '?' THEN
      BEGIN
        ILINE;
        CURS:=1;
        FOR I:=1 TO 100 DO
          BUF[I]:=' ';
        I:=1;
        WHILE NOT PEOS(I) DO
          BEGIN
            GETCHRR(BUF[I]);
            I:=I+1;
            END;
            (* WHILE *)
        END;
      (*IF BUF = '?' *)
    WHILE(BUF[CURS]=' ') AND (BUF[CURS]<>'?') DO
      CURS:=CURS+1;
    IF BUF[CURS]='?' THEN
      GOTO 1;
    CTYPE := DELIMTP;
    FCURS := CURS;
2:  IF (BUF[CURS]<='Z')AND(BUF[CURS]>='A') THEN
      BEGIN
        CTYPE:=DESCTP;
        LCURS:=CURS;
        CURS:=CURS+1;
        GOTO 2;
        END;
    IF (BUF[CURS]>='0')AND(BUF[CURS]<='9') THEN
      BEGIN
        IF NOT (BUF[FCURS] IN ['A'..'Z']) THEN
          CTYPE := DIGITTP
          ELSE
          CTYPE := DUMMYTP;
        CURS:=CURS+1;
        GOTO 2;
        END;
    ERR:=0;
    CASE CTYPE OF
0 :   BEGIN
        CTYPE:=ORD(BUF[CURS]);
        CURS:=CURS+1;
        END;
2 :   BEGIN
        FINDROW(FCURS,CURS-1,I);
        IF I<>0 THEN
          SROW:=I
          ELSE
          BEGIN
                  (* FIND ASSOC FN IN SYMTAB *)
            FINDROW(FCURS,LCURS,I);
            IF I<>0 THEN
              BEGIN
                S.NELT:=S.NELT+1;
                SROW:=S.NELT;
                FOR J:=1 TO 10 DO
                  S.NAME[S.NELT,J]:=' ';
                FOR J:=FCURS TO CURS-1 DO
                  S.NAME[S.NELT,J-FCURS+1]:=BUF[J];
                S.DPNO[SROW]:=I;
                END
                  (*I<>0*)
            ELSE
            BEGIN
              FIXSYM(FCURS,LCURS);
              GOTO 1;
              END;
            END;
            (* IF I<> 0 ELSE *)
        END;
         (*CASE DUMYTP *)
1 :   BEGIN
        FINDROW(FCURS,CURS-1,I);
        IF I=0 THEN
```

```
          BEGIN
            FIXSYM(FCURS,CURS-1);
            GOTO 1;
            END
        ELSE
        SROW:=I;
        END;
         (*CASE DESCTP *)
3:      BEGIN
        SROW:=0;
        FOR I:=FCURS TO CURS-1 DO
          SROW:=SROW*10+ ORD(BUF[I])-ORD('0');
        END
      END;
      (* CASE STMT *)
    CASE ERR OF
1:      WRITELN(OUTPUT,'INVALID CHARACTER');
0:      END;
      (*CASE STMT*)
    IF TRACE>2 THEN
      WRITELN(OUTPUT,'LEAVING TOKEN',CURS,CTYPE,SROW,ERR);
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                               VLINT(
         PARSE A VL2 EXPRESSION AND PERFORM SEMANTIC ACTIONS AS REQUIRED
     TO FORM A GRAPH.  ADD ENTRIES TO SYMBOL TABLE AND GRAPH STRUCTURE.
     PSTK IS THE STACK OF NONTERMINALS TRIED ALREADY
     VSTK IS A STACK OF VALUE SETS FOR REFERENCES
     FSTK IS A STACK OF DESRIPTORS AND DUMMY VARIABLES
     SSTK IS THE TOP DOWN PARSE OF THE EXPRESSION SO FAR
     EACH TOKEN FROM THE TOKEN ROUTINE IS MATCHED WITH AN ELEMENT
     IN A ROW OF THE PARSE TABLE (IN THIS TABLE, POS NUMBERS ARE
     TERMINALS, NEG ARE NONTERMINALS, POS NUMBERS MATCH THE NUMBER
     RETURNED BY TOKEN, NEG NUMBERS SPECIFY WHICH ROW OF THE PARSE
     TABLE TO PARSE NEXT).
          IF AN ELEMENT MATCHES, IT IS PLACED ON SSTK, IF IT IS AT
     THE END OF A ROW (PRODUCTION), THEN THE ELEMENTS OF SSTK ARE REPLACED
     BY -ROW' OF THE MATCH, PSTK IS POPPED AND THE CURRENT ROW IS THE
     TOP ELEMENT OF PSTK (ALONG WITH THE COLUMN POINTER IN PSTK).
     WHEN YOU GETTO THE BOTTOM OF PSTK, THEN YOURE DONE.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
   (* TOKEN *)
  PROCEDURE VLINT;
  LABEL 11,10,1,2,3,4,5,98,99;
  VAR VTOP,FTOP,STOP,PTOP,PROD,LOC,CURS,CTYPE
,SROW,GDESC ,GTOP, I,J,K,L,ANO :INTEGER;
  VSTK : ARRAY[1..GSIZE] OF VALTP;
  TRACE : INTEGER;
  FSTK,SSTK:ARRAY[1..150] OF INTEGER;
  PSTK:ARRAY[1..200] OF INTEGER;
  BUF:ARRAY[1..101] OF CHAR;
  DONE : BOOLEAN;
  PROCEDURE PROCESS(VAR DONE:BOOLEAN);
  LABEL 1,2,3,4,5;
  VAR I,J:INTEGER;
  PROCEDURE DUMPROC;
  BEGIN
    CASE PTBL.SRULE[-PROD] OF
19:     BEGIN
        VTOP:=VTOP+1;
        VSTK[VTOP]:=[0..MNVAL];
        END;
18:     BEGIN
        G.VAL[FSTK[1]]:=[1];
        FTOP:=FTOP-1;
        IF S.MVAL[ABS(G.PNO[FSTK[1]]) ]>1 THEN
          S.MVAL[ABS(G.PNO[FSTK[1]])  ]:=1;
        IF S.NVAL[ABS(G.PNO[FSTK[1]]) ]<1 THEN
          S.NVAL[ABS(G.PNO[FSTK[1]])  ]:=1;
        S.EVAL[ABS(G.PNO[FSTK[1]]) ]:=S.NVAL[ABS(G.PNO[FSTK[1]]) ];
        END;
```

```
17:   BEGIN
        GTOP:=GTOP+1;
        FSTK[1]:=GTOP;
        FTOP:=FTOP+1;
        ANO:=0;
        G.PNO[GTOP]:=S.PNO[SROW];
        G.DUMNUM[GTOP]:=SROW;
        G.VBL[GTOP]:=FALSE;
        G.ORDIRR[GTOP]:=FALSE;
        G.VAL[GTOP]:=[0..MNVAL];
        IF CHRR='E' THEN
          S.VTYPE[G.PNO[GTOP]]:=3;
        END;
         (* DIGIT *)
16:   BEGIN
            (* PUSH DIGIT ON STK *)
        FTOP := FTOP+1;
        FSTK[FTOP]:=-SROW;
        IF SROW>S.EVAL[G.PNO[FSTK[1]]] THEN
          S.EVAL[G.PNO[FSTK[1]]]:=SROW;
        IF CHRR<>'E' THEN
          IF SROW>S.NVAL[G.PNO[FSTK[1]]] THEN
            S.NVAL[G.PNO[FSTK[1]]]:=SROW;
        IF SROW<S.MVAL[G.PNO[FSTK[1]]] THEN
          S.MVAL[G.PNO[FSTK[1]]]:=SROW;
        END
      END;
      (*CASE*)
    END;
   (*DUMPROC*)
  BEGIN
    DONE:=FALSE;
    CASE PTBL.SRULE[-PROD] OF
         (* DESC *)
         (* SROW HAS LOC IN STAB OF DESC,ALOC NODE FOR DESC
          *)
         (* DUMMY *)
16,17,18,19:DUMPROC;
15:   BEGIN
            (* FIND DUMMY IN GRAPH, PUSH LOC IN GRAPH *)
        IF CHRR<>'E' THEN
          FOR I:=1 TO GTOP DO
            IF G.DUMNUM[I]=SROW THEN
              GOTO 3;
        GTOP:=GTOP+1;
        I:=GTOP;
        G.DUMNUM[I]:=SROW;
        G.PNO[I]:=S.DPNO[SROW];
        G.VBL[I]:=TRUE;
        G.ORDIRR[GTOP]:=TRUE;
        G.VAL[I]:=[0..MNVAL];
3:      FTOP:=FTOP+1;
        FSTK[FTOP]:=I;
        ANO:=ANO+1;
        IF CHRR='E' THEN
          S.VTYPE[G.PNO[FSTK[1]]]:=3;
        END;
         (* AREST *)
         (* POP VALUE AND DUMMY VAR, FIND DUMMY VAR IN G,SE
          T ARG*)
14,13:BEGIN
        G.VAL[FSTK[FTOP]]:=VSTK[VTOP];
        VTOP:=VTOP-1;
        FTOP:=FTOP-1;
        END;
         (* ALIST *)
         (* LINK DUMMY ON STK TO G DESC, J IS DUMMY DESC LO
          C*)
20,12,11:BEGIN
        J:=FSTK[FTOP];
        IF PTBL.SRULE[-PROD]=20 THEN
```

```
          BEGIN
                  (* G.PNO[FSTK[1]]:= -ABS(G.PNO[FSTK[1]]);*)
            G.ORDIRR[FSTK[1]]:=TRUE;
            END;
        G.LNK[FSTK[1],ANO]:=J;
        IF PTBL.SRULE[-PROD]<>20 THEN
          IF S.NARG[G.PNO[FSTK[1]]]<ANO THEN
            S.NARG[G.PNO[FSTK[1]]]:=ANO;
        ANO:=ANO-1;
        FTOP:=FTOP-1;
        FOR I:=1 TO MNVAL DO
          IF G.LNK[J,I]=0 THEN
            GOTO 5;
5:      G.LNK[J,I]:=FSTK[1];
        END;
         (* RNG *)
10:   BEGIN
            (*ALLOCATE NEW VAL ELT, PUT DIGIT IN THIS *)
        VTOP:=VTOP+1;
        VSTK[VTOP]:=[-FSTK[FTOP]];
        FTOP:=FTOP-1;
        END;
         (*RNG  *)
9:    BEGIN
            (* INTERVAL VARIABLE *)
        S.VTYPE[G.PNO[FSTK[1]]]:=2;
        VTOP:=VTOP+1;
        VSTK[VTOP]:=[ ];
        FOR I:=-FSTK[FTOP-1] TO -FSTK[FTOP] DO
          VSTK[VTOP]:=VSTK[VTOP]+[I];
        FTOP:=FTOP-2;
        END;
         (*INTERVAL VARIABLE**)
8:    BEGIN
            (* PUT DIGIT IN THE VAL SET *)
        VSTK[VTOP]:=VSTK[VTOP]+[-FSTK[FTOP]];
        FTOP:=FTOP-1;
        END;
         (* SEL *)
6,7:  BEGIN
            (* PUT VAL IN FSTK[1], PLACE IN G *)
        G.VAL[FSTK[1]]:=VSTK[VTOP];
        VTOP:=VTOP-1;
        FTOP:=FTOP-1;
        END;
         (* VLFORM *)
5,4:  BEGIN
            (* NOTHING *)
        ;
        END;
         (* ERULE *)
3:    BEGIN
            (* FIX SET OF THE GRAPH *)
        G.ESET:=G.VAL[FSTK[1]];
        FOR J:=1 TO 10 DO
          IF J IN G.ESET THEN
            K:=J;
        FS:=K;
        DONE:=TRUE;
        GOTO 2;
        END;
         (*VVERULE *)
2:    ;
         (*VVERULE*)
1:    BEGIN
            (* POP DIGIT, PUT INTO GRAPH *)
        G.COEF:=-FSTK[1];
        END
      END;
      (* CASE STMT *)
2:  ;
```

```
    END;
   (*PROCESS*)
  BEGIN
    IF INFILE = 0 THEN
      WRITELN(OFILE,'RULE     ',G.RNO:5);
    FOR I:=1 TO GSIZE DO
      G.PNO[I]:=0;
    FOR I:=1 TO GSIZE DO
      FOR J:=1 TO MLNK DO
        G.LNK[I,J]:=0;
    CURS:=101;
    BUF[101]:='?';
    TRACE:=2;
    FOR I:=1 TO 150 DO
      SSTK[I]:=0;
    VTOP:=0;
    FTOP:=0;
    GTOP:=0;
    STOP:=1;
    PTOP:=0;
11: PROD:=-1;
    LOC:=1;
    WITH PTBL DO
      BEGIN
1:      IF SSTK[STOP]=0 THEN
          BEGIN
            TOKEN(CURS,CTYPE,SROW,ERR,BUF);
            SSTK[STOP]:=CTYPE;
            IF ERR <> 0 THEN
              GOTO 99;
            END;
        IF (RHS[-PROD,LOC]<0) AND (RHS[-PROD,LOC]<>SSTK[STOP]) THEN
          BEGIN
                   (* PUSH PROD AND LOC *)
            PSTK[PTOP+1]:=PROD;
            PSTK[PTOP+2]:=LOC;
            IF TRACE>2 THEN
              WRITELN(OUTPUT,'PUSH',PROD,LOC);
            PTOP:=PTOP+2;
            PROD:=RHS[-PROD,LOC];
            LOC:=1;
            GOTO 1;
            END;
            (* IF --- AND --- THEN*)
        IF RHS[-PROD,LOC]<>0 THEN
          IF RHS[-PROD,LOC]=SSTK[STOP] THEN
            BEGIN
                      (* ENTRY IN PT MATCHES TOKEN *)
              STOP:=STOP+1;
              LOC:=LOC+1;
              GOTO 1;
              END
            (* RHS = SSTK *)
        ELSE
        BEGIN
               (* ENTRY DOES NOT MATCH SSTK*)
          STOP:=STOP-(LOC-1);
          PROD:=PROD-1;
          LOC:=1;
          IF TRACE>2 THEN
            WRITELN(OUTPUT,'NOMATCH',PROD);
10:       IF CONT[-PROD] THEN
            GOTO 1
            ELSE
            BEGIN
              PTOP:=PTOP-2;
              IF PTOP=-2 THEN
                GOTO 98;
              STOP:=STOP-(PSTK[PTOP+2]-1);
              PROD:=PSTK[PTOP+1]-1;
              GOTO 10;
              END;
          END;
            (* IF RHS = SSTK *)
        BEGIN
               (* EXECUTE PROC *)
          PROCESS(DONE);
          IF DONE THEN
            GOTO 2;
          IF TRACE>2 THEN
            WRITELN(OUTPUT,'PROC',PROD);
               (*REPLACE LOC-1 ENTRIES IN SSTK WITH PROD *)
          STOP:=STOP-(LOC-1);
          FOR J:=STOP+1 TO 150 DO
            IF J+LOC-2<=150 THEN
              SSTK[J]:=SSTK[J+LOC-2];
          WHILE CONT[-PROD] DO
            PROD:=PROD+1;
          SSTK[STOP]:=PROD;
          PTOP:=PTOP-2;
          IF PTOP=-2 THEN
            GOTO 2;
          PROD:=PSTK[PTOP+1];
          LOC:=PSTK[PTOP+2]+1;
          IF TRACE>2 THEN
            WRITELN(OUTPUT,'POP',PROD,LOC,STOP);
          STOP:=STOP+1;
          GOTO 1;
          END;
        END;
      (*WITH*)
    GOTO 99;
98: WRITELN(OFILE,'INVALID SYNTAX',CTYPE,'EXPECTING ',PTBL.RHS[-PROD,LOC]);
    ERR:=1;
    IF CTYPE <= 2 THEN
      GOTO 99;
    ERR:=0;
    STOP:=1;
    WHILE SSTK[STOP+1]<>0 DO
      BEGIN
        IF SSTK[STOP]<0 THEN
          WHILE PTBL.CONT[-SSTK[STOP]] DO
            SSTK[STOP]:=SSTK[STOP]+1;
        STOP:=STOP+1;
        END;
    SSTK[STOP]:=0;
    FOR J:=1 TO CURS-1 DO
      WRITE(OFILE,BUF[J]);
    PUTSEG(OFILE);
    WRITE(OFILE,'RETYPE LAST CHARACTER');
    PUTSEG(OFILE);
    ILINE;
    READ(IFILE,BUF[CURS-1]);
    CURS:=CURS-1;
    I:=1;
    WHILE NOT EOLN(IFILE) DO
      BEGIN
        FOR J:=CURS+I TO 99 DO
          BUF[J+1]:=BUF[J];
        READ(IFILE,BUF[CURS+I]);
        I:=I+1;
        END;
    PTOP:=0;
    STOP:=1;
    GOTO 11;
    GOTO 99;
2:  IF PSTK[1] < -3 THEN
      GOTO 98;
     (* IF RESTRICTIN, THEN PLACE CONS AT;
    END
  OF G AND DELETE INCOMMING LINKS*)  IF CHAR='R' THEN BEGIN I:=1;
  WHILE G.LNK[GTOP,I]<>0 DO
```

```
    BEGIN
      G.LNK[GSIZE,I]:=G.LNK[GTOP,I];
      J:=1;
      WHILE G.LNK[G.LNK[GTOP,I],J]<>0 DO
        J:=J+1;
      G.LNK[G.LNK[GTOP,I],J-1]:=0;
      G.LNK[GTOP,I]:=0;
      I:=I+1;
      END;
   (*WHILE G...<>0*)
  G.VBL[GSIZE]:=G.VBL[GTOP];
  G.ORDIRR[GSIZE]:=G.ORDIRR[GTOP];
  G.VAL[GSIZE]:=G.VAL[GTOP];
  G.PNO[GSIZE]:=G.PNO[GTOP];
  END;
(*IF CHPR='R'*)
99:;
END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                               COSTG(P:GPTR;
             EVALUATE THE COST OF THIS GRAPH (COST FUNCTION CT).
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
(* VLINT *)
PROCEDURE COSTG(P:GPTR;
CT:INTEGER);
LABEL 6;
VAR J,I:INTEGER;
INSD,CTNEG:BOOLEAN;
Q:GPTR;
BEGIN
(*COSTG*)
IF CT<0 THEN
  CTNEG:=TRUE
   ELSE
  CTNEG:=FALSE;
CT:=ABS(CT);
IF CT IN [1,2,3,4] THEN
  CASE CT OF
1,3:BEGIN
      CASE CT OF
1:        INSD:=TRUE;
3:        INSD:=FALSE
          END;
         (*CASE STMT*)
      P.COST[CT]:=0;
      Q:=GSET;
      WHILE Q<>NIL DO
        BEGIN
          IF (CT=1)AND(ES IN Q.ESET)AND(Q.FP)
OR( CT=3)AND (NOT (ES IN Q.ESET)) THEN
            IF SUBG1(P,Q,0,AQP.FREEC,TRUE) THEN
              P.COST[CT]:=P.COST[CT]+1;
          Q:=Q.NXTN;
          END;
         (*WHILE Q<>NIL*)
      IF CT=3 THEN
        END;
         (*CASE 1*)
2,4:  BEGIN
        P.COST[CT]:=0;
        FOR J:=1 TO GSIZE DO
          IF P.LNK[J,1]<>0 THEN
            IF NOT P.VBL[J] THEN
              IF (S.NARG[ABS(P.PNO[J])]>1)OR(P.VAL[J]<>[0..MNVAL]) THEN
                IF CT=2 THEN
                  P.COST[2]:=P.COST[2]+1
                  ELSE
                  P.COST[4]:=P.COST[4]+S.VCOST[ABS(P.PNO[J])];
        IF P.MSEL<>NIL THEN
          FOR J:=1 TO MST.NMST DO
            IF P.MSEL.CVAL[MST.PTR[J]]<>[0..MNVAL] THEN
```

```
              IF CT=2 THEN
                P.COST[ 2 ]:=P.COST[ 2 ]+1
                ELSE
                P.COST[ 4 ]:=P.COST[4 ]+ S.VCOST[ABS(MST.PNO[MST.PTR[ J ]]) ];
        END
         (*CASE 2*)
      END;
      (*CASE STMT*)
    IF CTNEG THEN
      P.COST[ CT ]:=-P.COST[ CT ];
    END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                                TRIMG(VAR NSTAR:GPTR;
            TRIM A LIST OF GRAPHS TO MAYS GRAPHS ACCORDING TO FUNCTIONAL
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
(*COSTG*)
PROCEDURE TRIMG(VAR NSTAR:GPTR;
MAXS:INTEGER);
LABEL 1,2,99;
TYPE ATYPE = ARRAY[0..300] OF GPTR;
VAR CA:ATYPE;
X:REAL;
P,Q:GPTR;
NC,I,J,IB,IC:INTEGER;
BEGIN
   (*TRIMG*)
  IC:=1;
  IB:=1;
  P:=NSTAR;
  NC:=0;
  WHILE P<>NIL DO
    BEGIN
      Q:=P;
      IF P.FP THEN
        BEGIN
          IF P.COST[ 3 ]=0 THEN
            BEGIN
              NEWG(Q);
              Q:=P;
              P.RNO:=CRULENO-1;
              Q.NXTN:=MQ;
              MQ:=Q;
              NMQ:=NMQ+1;
              END;
          NC:=NC+1;
          CA[ NC ]:=P;
          P:=P.NXTN;
          END
         (*IF THEN*)
      ELSE
         (*FP FALSE*)
      BEGIN
        IF P.COST[ 3 ]=100 THEN
          BEGIN
            P.MSEL.NXTC:=AQP.FREEC;
            AQP.FREEC:=P.MSEL;
            P.MSEL:=NIL;
            END;
        P:=P.NXTN;
        Q.NXTN:=FREEG;
        FREEG:=Q;
        END;
      END;
   (*WHILE P<>NIL *)
  CA[ NC+1 ]:=CA[ NC ];
  CA[ 0 ]:=CA[ 1 ];
1:IF NC<=MAXS THEN
    GOTO 99;
  I:=1;
  IF MAXS=0 THEN
    GOTO 2;
   FOR I:=IB TO NC-IB DO
      FOR J:=I+IB TO NC DO
         IF CA[J].COST[ABS(PRM.CSTF[IC])] < CA[I].COST[ABS(PRM.CSTF[IC])] THEN
            BEGIN
               P:=CA[J];
               CA[J]:=CA[I];
               CA[I]:=P;
               END;
   I:=MAXS+1;
   IF PRM.TOLER[IC]=TFUNC(PRM.TOLER[IC]) THEN
      X:=PRM.TOLER[IC]
      ELSE
      X:=PRM.TOLER[IC]*(CA[NC].COST[ABS(PRM.CSTF
[IC])]-CA[1].COST[ABS(PRM.CSTF [IC])]) ;
   IF IC<>PRM.NF THEN
      WHILE (CA[MAXS].COST[ABS(PRM.CSTF[IC])]
>= CA[I].COST[ABS(PRM.CSTF[IC])] - X) AND (I<=NC) DO
         I:=I+1;
     (* RETURN ELEMENTS FROM I TO NC*)
2:FOR J:=I TO NC DO
      BEGIN
         CA[J].NXTN:=FRFEG;
         FRFEG:=CA[J];
         END;
   NC:=I-1;
   IB:=MAXS-1;
   WHILE (CA[MAXS].COST[ABS(PRM.CSTF[IC])] <= CA
[IB].COST[ABS(PRM.CSTF[IC])] + X) AND (IB>0) DO
      IB:=IB-1;
   IB:=IB+1;
   IC:=IC+1;
   IF IC<=PRM.NF THEN
      GOTO 1;
99:NSTAR:=NIL;
   FOR I:=1 TO NC DO
      BEGIN
         CA[I].NXTN:=NSTAR;
         NSTAR:=CA[I];
         END;
   END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                                                COMPMS
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
(*TRIMG*)
PROCEDURE COMPMS(GSET:GPTR;
FS,MPNO,VALUE:INTEGER;
VAR NPT1:IARRAY;
VAR NPT2:IARRAY;
VAR FA1:INTEGER;
VAR FA2:INTEGER);
VAR G:GPTR;
I,J,K,L,M,N:INTEGER;
BEGIN
    (* INPUT LIST OF EVENTS,  FNCTNS AND VAUES.  CALCU
   LATE META SELECTORS NPT AND FORALL; ADD TO EVENT*)
    (* ADD INFO TO MST *)
   FOR I:=1 TO 2 DO
      BEGIN
         MST.PNO[MST.PTR[MST.NMST+I]]:=MPNO;
         MST.VAL[MST.PTR[MST.NMST+I]]:=VALUE;
         MST.SYMPTR[MST.PTR[MST.NMST+I]]:=I;
         END;
   MST.NMST:=MST.NMST+2;
   N:=MST.NMST;
   G:=GSET;
   NF1:=0;
   WHILE G<>NIL DO
      BEGIN
         K:=0;
         L:=0;
         FOR I:=1 TO GSIZE DO
```

```
        IF G.PNO[I]=MST.PNO[MST.PTR[N]] THEN
          BEGIN
            K:=K+1;
            IF MST.VAL[MST.PTR[N]] IN G.VAL[I] THEN
              L:=L+1;
            END;
      G.MSEL.CVAL[MST.PTR[N]]:=[L];
      IF L>S.NVAL[MST.SYMPTR[MST.PTR[N]]] THEN
        BEGIN
          S.NVAL[MST.SYMPTR[MST.PTR[N]]]:=L;
          S.FVAL[MST.SYMPTR[MST.PTR[N]]]:=L;
          END;
      IF K=L THEN
        G.MSEL.CVAL[MST.PTR[N-1]]:=[1]
        ELSE
        G.MSEL.CVAL[MST.PTR[N-1]]:=[0];
      IF K=L THEN
        IF ES IN G.ESET THEN
          FA1:=FA1+1
          ELSE
          FA0:=FA0+1;
      IF ES IN G.FSET THEN
        NF1:=NF1+1;
      IF ES IN G.ESET THEN
        NPT1[L]:=NPT1[L]+1
        ELSE
        NPT0[L]:=NPT0[L]+1;
      G:=G.NXTN;
      END;
   (*WHILE*)
  END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                           TRIMM
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
(*COMPMS*)
PROCEDURE TRIMM;
VAR I,J,K,L,M:INTEGER;
G:GPTR;
BEGIN
  IF MST.METATRIM<MST.NMST THEN
    BEGIN
      FOR I:=1 TO MST.METATRIM DO
        FOR J:=I+1 TO MST.NMST DO
          IF (MST.F1COV[MST.PTR[I]]<MST.F1COV
          [MST.PTR[J]])OR (MST.F1COV[MST.PTR[I
          ]]=MST.F1COV[MST.PTR[J]]) AND (MST.F0COV
[MST.PTR[I]]>MST.F0COV[MST.PTR[J]]) THEN
            BEGIN
              L:=MST.PTR[I];
              MST.PTR[I]:=MST.PTR[J];
              MST.PTR[J]:=L;
              END;
      MST.NMST:=MST.METATRIM;
      END;
  END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                         ADDMETA
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
(*TRIMM*)
PROCEDURE ADDMETA;
(* THIS PROCEDURE CALCULATES A SET OF META SELECTO
RS AND HAS THEM LOADED IN TO THE EVENT *)
VAR I,J,K,L,FA1,FA0:INTEGER;
NPT1,NPT0:IARRAY;
BEGIN
  FOR I:=1 TO SYMSZE DO
    IF (S.NARG[I]=1)AND(S.NAME[I,4]<>'-') THEN
      FOR J:=S.MVAL[I] TO S.NVAL[I] DO
        BEGIN
          FOR L:=0 TO MNVAL DO
            BEGIN
```

```
              NPT1[ L ]:=0;
              NPT0[ L ]:=0;
              END;
          FA1:=0;
          FA0:=0;
          COMPMS(GSET,ES,I,J,NPT1,NPT0,FA1,FA0);
          MST.F1COV[MST.PTR[MST.NMST-1]]:=FA1;
          MST.F0COV[MST.PTR[MST.NMST-1]]:=FA0;
          K:=-1000;
          FOR L:=0 TO MNVAL DO
            IF NPT1[L]>K THEN
              BEGIN
                K:=NPT1[L];
                FA1:=L;
                END;
          MST.F1COV[MST.PTR[MST.NMST]]:=K;
          MST.F0COV[MST.PTR[MST.NMST]]:=NPT0[FA1];
          END;
  TRIMM;
  IF 6 IN TRACE THEN
    BEGIN
      EXPLN(6);
      PMETAD;
      END;
   (*IF TRACE*)
  END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                         ADDML
      ADD THE MOST AND LEAST PARTS OF 2-ARY FNCTNS
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
(*ADDMETA*)
PROCEDURE ADDML;
LABEL 2;
(* SELECT ONE PREDICATE AND ADD LEFT AND RIGHT
END
S TO STABLE THEN ADD THE LEFT OR RIGHT
END
FNCTN TO THE GRAPH FOR EACH EVENT*) VAR I,J,K,L,M:INTEGER;
G:GPTR;
BEGIN
FOR I:=1 TO S.NELT DO
IF (S.NARG[I]=2)AND(S.MVAL[I]=1) THEN
  BEGIN
      (*ADD TO STABLE*)
    S.NELT:=S.NELT+2;
    S.NAME[S.NELT-1]:='MST-      ';
    S.NAME[S.NELT]:='LST-      ';
    FOR K:=S.NELT-1 TO S.NELT DO
      BEGIN
        FOR J:=5 TO 10 DO
          S.NAME[K,J]:=S.NAME[I,J-4];
        S.PNO[K]:=K;
        S.NARG[K]:=1;
        S.NVAL[K]:=1;
        S.MVAL[K]:=1;
        S.FVAL[K]:=1;
        END;
      (*FOR K*)
    G:=GSET;
    WHILE G<>NIL DO
      BEGIN
        FOR J:=1 TO GSIZE DO
          IF G.PNO[J]=I THEN
            IF G.PNO[G.LNK[J,1]]=G.PNO[G.LNK[J,2]] THEN
              FOR K:=1 TO 2 DO
                BEGIN
                  M:=1;
                  WHILE G.LNK[G.LNK[J,K],M]<>0 DO
                    BEGIN
                      L:=G.LNK[G.LNK[J,K],M];
                      IF (G.PNO[L]=I)AND(J<>L) THEN
```

```
                        IF  (L<J)OR(G.LNK[L,K]<>G.LNK[J,K])  THEN
                          GOTO 2
                          ELSE
                          M:=M+1
                          ELSE
                          M:=M+1;
                      END;
                            (*ADD NODE TO GRAPH*)
                  L:=1;
                  WHILE G.LNK[L,1]<>0 DO
                    L:=L+1;
                  G.PNO[L]:=S.NELT-2+K;
                  G.VBL[L]:=FALSE;
                  G.ORDIRR[L]:=FALSE;
                  G.VAL[L]:=[1];
                  G.LNK[L,1]:=G.LNK[J,K];
                  G.LNK[G.LNK[J,K],M]:=L;
2:                END;
            (*FOR K*)
        G:=G.NXTN;
        END;
      (*WHILE*)
    END;
(*FOR I*)
END;
(*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                                  COVER(ES:INTEGER);
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*)
(*ADDML*)
PROCEDURE COVER(VAR ES:INTEGER);
LABEL 1,2;
VAR G,Q,P,OFSTAR:GPTR;
I,J,K:INTEGER;
PROCEDURE ABSOURB(STAR:GPTR);
BEGIN
P:=STAR;
WHILE P<>NIL DO
  BEGIN
    Q:=P.NXTN;
    WHILE Q<>NIL DO
      BEGIN
        IF SUBG1(P,Q,C,AQP.FREEC,TRUE) THEN
          IF SUBG1(Q,P,C,AQP.FREEC,TRUE) THEN
            P.FP:=FALSE;
        Q:=Q.NXTN;
        END;
    P:=P.NXTN;
    END;
END;
(*ABSOURB*)
BEGIN
WRITELN(OFILE,'ENTER DECISION NUMBER OF SET TO BE COVERED');
PUTSEG(OFILE);
PUTSEG(OFILE);
GETSEG(IFILE);
WHILE EOLN(IFILE) DO
  GETSEG(IFILE);
READ(IFILE,ES);
MST.MMST:=0;
IF MST.METATRIM<>0 THEN
  ADDMETA;
G:=COVSET;
IF G<>NIL THEN
  BEGIN
    WHILE G.NXTN<>NIL DO
      G:=G.NXTN;
    G.NXTN:=FREEG;
    FREEG:=COVSET;
    END;
COVSET:=NIL;
G:=GSET;
```

```
WHILE G<>NIL DO
  BEGIN
    G.FP:=TRUE;
    G:=G.NXTN;
    END;
G:=GSET;
WHILE G<>NIL DO
  BEGIN
    IF G.FP AND (ES IN G.ESET) THEN
      BEGIN
        FOR I:=1 TO GSIZE DO
          G.COUNT[I]:=0;
        MQ:=NIL;
        PSTAR:=NIL;
        STAR:=NIL;
        NMQ:=0;
            (*SET UP INITIAL STAR*)
        IF 10 IN TRACE THEN
          BEGIN
            WRITELN(OFILE,'NOW COVERING EVENT');
            PGRAPH(G,S);
            EXPLN(10);
            END;
        FOR I:=1 TO GSIZE DO
          IF (NOT G.VBL[I])AND(G.LNK[I, 1]<>0)AND( G.LNK[I, 2]=0) THEN
            BEGIN
              NEWG(G1);
              J:=G1.RNO;
              G1:=G;
              G1.COUNT[I]:=1;
              G1.RNO:=J;
              G1.NXTN:=STAR;
              STAR:=G1;
              FOR K:=1 TO GSIZE DO
                FOR J:=1 TO MLNK DO
                  G1.LNK[K,J]:=0;
              J:=1;
              WHILE G.LNK[I,J]<>0 DO
                BEGIN
                  G1.LNK[I,J]:=G.LNK[I,J];
                  G1.LNK[G1.LNK[I,J],1]:=I;
                  G1.COUNT[G1.LNK[I,J]]:=1;
                  J:=J+1;
                  END;
              END;
2:      G1:=STAR;
        IF 1 IN TRACE THEN
          BEGIN
            WRITELN(OFILE, 'THE FOLLOWING FORMULAS',
 ' ARE IN THE UNTRIMMED STAR');
            EXPLN(0);
            END;
        WHILE G1<>NIL DO
          BEGIN
            FOR J:=1 TO PRM.NF DO
              COSTG(G1,PRM.CSTF[J]);
              G1.FP:=TRUE;
            IF 1 IN TRACE THEN
              PGRAPH(G1,S);
            G1:=G1.NXTN;
            END;
            (*ABSOURPTION *)
        ABSOURB(STAR);
        TRIMG(STAR,PRM.MAXSTAR);
        IF 1 IN TRACE THEN
          BEGIN
            WRITELN(OFILE, 'THE FOLLOWING FORMULAS REMAIN', ' AFTER TRIMMING');
            EXPLN(1);
            END;
        IF (NMQ>=PRM.NCONSIST)OR (STAR=NIL) THEN
          GOTO 1;
```

```
       G1:=STAR;
       WHILE G1<>NIL DO
         BEGIN
           OPSTAR:=PSTAR;
           NEWGP(PRM.ALTER,G1,G,PSTAR);
           IF 1 IN TRACE THEN
             PGRAPH(G1,S);
                 (*ABSOURPTION *)
           P:=PSTAR;
           WHILE P<>OPSTAR DO
             BEGIN
               Q:=OPSTAR;
               WHILE Q<>NIL DO
                 BEGIN
                   IF SUBG1(P,Q,0,AQP.FREEC,TRUE) THEN
                     Q.FP:=FALSE
                     ELSE
                     IF SUBG1(Q,P,0,AQP.FREEC,TRUE) THEN
                       P.FP:=FALSE;
                   Q:=Q.NXTN;
                   END;
               P:=P.NXTN;
               END;
           G1:=G1.NXTN;
           END;
           (*WHILE G1<>NIL*)
           (* RETURN CURRENT STAR TO FREE LIST*)
       G1:=STAR;
       WHILE G1.NXTN<>NIL DO
         G1:=G1.NXTN;
       G1.NXTN:=FREEG;
       FREEG:=STAR;
       STAR:=PSTAR;
       PSTAR:=NIL;
       GOTO 2;
       (*NOW HAVE MQ LIST OF CONSITENT FORMULAS;
       APPLY AQ PROC*) 1: G1:=MQ;
       IF 2 IN TRACE THEN
         BEGIN
           WRITELN(OFILE,'THE CONSISTENT FORMULAS:');
           EXPLN(2);
           END;
       WHILE G1<>NIL DO
         BEGIN
           IF 2 IN TRACE THEN
             BEGIN
               WRITELN(OFILE,'BEFORE AQ:');
               PGRAPH(G1,S);
               END;
           AQSET(GSET,ES,G1);
           FOR I:=1 TO PRM.NF DO
             IF PRM.CSTF[I]<>-3 THEN
               COSTG(G1,PRM.CSTF[I]);
           IF 2 IN TRACE THEN
             BEGIN
               WRITELN(OFILE,'AFTER AQ:');
               PGRAPH(G1,S);
               END;
           G1:=G1.NXTN;
           END;
       ABSOURB(MQ);
       G1:=MQ;
       IF 9 IN TRACE THEN
         WRITELN(OFILE,'THE FOLLOWING ARE ALTERNATIVE',
'CONSISTENT GENERALIZATIONS');
       WHILE G1<>NIL DO
         BEGIN
           IF 9 IN TRACE THEN
             IF G1.FP THEN
               PGRAPH(G1,S);
           G1.COST[3]:=-100;
```

```
            G1:=G1.NXTN;
            END;
        TRIMG(MQ,1);
        IF 3 IN TRACE THEN
          BEGIN
            WRITELN(CFILE,'THE SELECTED MQ FORMULA IS:');
            PGRAPH(MQ,S);
            EXPLN(3);
            END;
        MQ.NXTN:=COVSET;
        COVSET:=MQ;
        G1:=G;
        WHILE G1<>NIL DO
          BEGIN
            IF (ES IN G1.ESET)AND(G1.FP) THEN
              IF SUBG1(MQ,G1,0,AQP.FREEC,TRUE) THEN
                G1.FP:=FALSE;
            G1:=G1.NXTN;
            END;
        END;
      (*IF G.FP ETC*)
    G:=G.NXTN;
    END;
(*WHILE G<>*)
WRITELN(OFILE,'THE FOLLOWING FORMULAS COVER SET ',ES);
G:=COVSET;
WHILE G<>NIL DO
  BEGIN
    WRITELN(OFILE,'THIS RULE COVERS',-G.COST[1],' NEW RULES');
    PGRAPH(G,S);
    G:=G.NXTN;
    END;
IF MST.NMST<>0 THEN
  PMETAD;
END;
(*COVER*)
BEGIN
INIT;
RESET(GFILE);
WHILE NOT EOF(GFILE) DO
  BEGIN
    NEWG(G);
    GIN(G);
    G.NXTN:=GSET;
    GSET:=G;
    END;
(*WHILE*)
CHRR:=' ';
INFILE:=1;
RESET(CFILE);
WRITELN(OFILE,'ENTER ONE CHAR: (P)ARAMETERS, (V)L1, (C)OVER, ',
 '(M)ODIFY, (H)ELP FOR MORE OPTNS');
1:IF INFILE=0 THEN
  WRITELN(OFILE,'ENTER ONE CHAR: (P)ARAMETERS, (V)L1, (C)OVER, ',
 '(M)ODIFY, (H)ELP FOR MORE OPTNS');
PUTSEG(OFILE);
ILINE;
GETCHRR(CHRR);
IF CHRR IN ['R','Q','M','C','P','E','L','H','V','S'] THEN
  CASE CHRR OF
'H':BEGIN
      WRITELN(OFILE,'     READ IN RESTRICTIONS (R) ');
      WRITELN(OFILE,'     MODIFY RULES (M) ');
      WRITELN(OFILE,'     GET HELP (H) ');
      WRITELN(OFILE,'     INCLUDE MOST-LEAST TYPE SELECTORS (L)');
      WRITELN(OFILE,'     COVER SET OF RULES (C) ');
      WRITELN(OFILE,'     USE VL1 MODE (V) ');
      WRITELN(OFILE,'     MODIFY PARAMETERS (P) ');
      WRITELN(OFILE,'     ENTER DOMAIN STRUCTURE RULES (E)');
      WRITELN(OFILE,'     ADD EQUIV TYPE SEL (S) ');
      WRITELN(CFILE,'     QUIT (Q) ');
      IF PEOS(I) THEN
        CHRR:='H'
        ELSE
        WHILE NOT PEOS(I)  DO
          GETCHRR(CHRR);
      IF CHRR IN['R','M','C','P','L','E','V','S'] THEN
        CASE CHRR OF
'R':      EXPLN(21);
'M':      EXPLN(22);
'C':      EXPLN(23);
'P':      EXPLN(24);
'V':      EXPLN(27);
'E':      EXPLN(25);
'S':      EXPLN(29);
'L':      EXPLN(28)
          END
         (*CASE STMT*)
      ELSE
      EXPLN(26);
      PUTSEG(OFILE);
      END;
'P':BEGIN
      ENTERP;
      END;
      (*CASE P*)
'E':ENTERD;
'V':VL1;
'L':BEGIN
      ADDML;
      PRM.EXTMTY:=TRUE;
      END;
'S':BEGIN
      G:=GSET;
      WHILE G<>NIL DO
        BEGIN
          ADDSEL(G);
          G:=G.NXTN;
          END;
      PRM.EQUIV:=TRUE;
      END;
'R':BEGIN
      NEWG(G);
      VLINT(G,ERR,ES);
      G.NXTN:=RESTLIST;
      RESTLIST:=G;
      END;
      (*CASE R*)
'M':BEGIN
      IF INFILE=0 THEN
        WRITE(OFILE,'ADD OR DELETE RULE? ');
      PUTSEG(OFILE);
      ILINE;
      GETCHRR(CHRR1);
      IF CHRR1 IN ['A','D'] THEN
        CASE CHRR1 OF
'A':      BEGIN
            NEWG(G);
            ERR:=1;
            NEW(G.MSEL);
            WHILE ERR<>0 DO
              BEGIN
                ERR:=0;
                IF INFILE=0 THEN
                  WRITELN(OFILE,'ENTER RULE');
                PUTSEG(OFILE);
                VLINT(G,ERR,ES);
                END;
                  (*WHILE*)
            G.NXTN:=GSET;
            R:=RESTLIST;
            WHILE R<>NIL DO
```

```
              BEGIN
                IF SUBG1(R,G,1,AQP.FREEC,TRUE) THEN
                  ; R:=R.NXTN;
                END;
            GSET:=G;
            FND;
               (*CASE A*)
'D':      BEGIN
            G1:=GSET;
            WHILE G1<>NIL DO
              BEGIN
                WRITELN(OFILE,'DELETE THE FOLLOWING RULE? ');
                PUTSEG(OFILE);
                PGPAPH(G1,S);
                TLINE;
                GETCHRR(CHRR);
                IF CHRP = 'Y' THEN
                  BEGIN
                    G2:=G1.NXTN;
                    G1.NXTN:=FREEG;
                    FREEG:=G1;
                    IF G1=GSET THEN
                      GSET:=G2
                      ELSE
                      G.NXTN:=G2;
                    G1:=G2;
                    END;
                IF CHPR = 'N' THEN
                  BEGIN
                    G:=G1;
                    G1:=G1.NXTN;
                    END;
                IF CHRR = 'Q' THEN
                  GOTO 1;
                FND
                  (*WHILE*)
            ;
            END
               (*CASF D*)
          END;
         (*CASE STMT *)
      END;
      (*CASE M*)
'C':BEGIN
      COVEF(ES);
      END;
      (*CASE C*)
'Q':BEGIN
      GOTO 99;
      END
      (*CASE Q*)
    FND;
(* CASF STMT *)
GOTO 1;
99:END.
```

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-77-876	2.	3. Recipient's Accession No.

4. Title and Subtitle

INDUCE-1: An Interactive Inductive Inference Program in VL_{21} Logic System

5. Report Date
May 1977

6.

7. Author(s)
James B. Larson

8. Performing Organization Rept. No.

9. Performing Organization Name and Address
Department of Computer Science
University of Illinois
Urbana, Illinois 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.
NSF MCS 74-03514

12. Sponsoring Organization Name and Address
National Science Foundation
Washington, DC

13. Type of Report & Period Covered

14.

15. Supplementary Notes

16. Abstracts

The program INDUCE-1 is a PASCAL program which generalizes VL_{21} decision rules to form consistent, complete, and near optimal VL_2 decision rules under a user specified criterion. The paper supplies the list of parameters and commands for operating the program, the I/O files which the program uses, description of the program structure and a complete program listing.

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

VL_2
variable-valued logic
first order predicate logic
inductive inference
production systems
decision rules

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

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