1974-6

RSM

Documentation of OMNIBUS - A Utility Program for Quantization of Variables and Evaluation of Inductively Derived Logic Formulas

Internal Report

August 1974

This program can logically be divided into three sections:

1) Data base manipulation

2) Event/formula evaluation

3) Statistics generation

As much as possible, items will be clustered into these groups for clarity in what follows.

* *

Input/Output

Knowledge of PL1 input/output operations will be helpful in what follows but will hopefully not be necessary. The data base is logically divided into clusters of events called classes. Each class is in turn divided into ordered sets of variable values known as events. Each variable value is a real number. This program is generally oriented towards manipulation of blocks of events as opposed to individual events.

Classes are assumed to be numbered consecutively starting at zero (as are quantized variable values). The variables themselves are consecutively numbered starting at one.

The input can be logically divided into the following units:

1) Primary parameter input (get data)

2) Secondary parameter input (get data)

3). Title (get edit)

4) Quantization thresholds (get list)

5) Event input (get edit)

6) Parameter and formula specification (get list or edit)

Most of the above units are optional. In addition, it is not necessary that the last two be in the system input stream. Provision is made for iterative application (if the input stream is not empty upon completion of one pass, the program will reinitialize variables and process the next pass).

* * * * * * * * * Primary Parameter Input * * * * * * * * * *

The following items are read in PL1 data-directed input. Each parameter is an assignment statement of the form:

Parameter = Value

where parameter is any of the names given below. Value can be any integer, quoted string of alpha-numeric data or bit string (of the form '0110'B) depending on the parameter.

* A semi-colon (;) must follow the last

* input parameter.

$ $ $ $ $ $ $ $ $ File Specifications $ $ $ $ $ $ $ $ $ $

DDNAME

In the following items 'DONAME' represents the name on your DD card that describes the file to be used.

1) E_IN_FILE = 'DONAME' (DEFAULT = 'SYSIN') - file which contains the raw input events

2) E_OUT_FILE = 'DONAME' (NO DEFAULT) - file in which quantized and trimmed events are written

3) FORMULA_FILE = 'DONAME' (DEFAULT = 'SYSIN') - file which contains formula information

$ $ $ $ $ $ $ General Data Description Parameters $ $ $ $ $ $ $ $

The following parameters indicate how the job appears in the input stream before any processing.

represents any integer value

4) #_INEVENTS = # (do not include if CLASS_FORMAT is given) - specifies how many events are in the input stream. They are read in as a group regardless of class

5) CLASS_FORMAT = '#1,#2,#3,#4, #1,#2,#3,#4, ...' (do not include if #_INEVENTS is given) - specifies that event input consists of distinct classes. Each class has four (4) numbers associated with it:

   #1 - class number
   #2 - total events of this class in input
   #3 - position (relative to the first event of the class) at which events will begin being kept
   #4 - number of events starting at #3 to be kept

   Events not kept are never seen by the program. #3 may be one (beginning of class) and #4 may be the overall length of class. Classes are assigned in the order given here.

6) #_VAR = # number of variables per event

7) #_CLASSES=# total number of classes (must be a small positive number if class is not relevant to this run)

8) VAR_CARD = '#,#,#,#...' A list of #_VAR numbers each representing the cardinality of that variable assigned in order given)

9) FORMAT_E_IN = 'FIELD_WIDTH=#, #_PER_LINE = #' indicates field width (DEFAULT = 5) and number of variables per card (DEFAULT = 16) of input events (see programming note #18)

10) FORMAT_E_OUT = 'FIELD_WIDTH = #, #_PER_LINE = #' indicates field width and number of variables per

card on output of processed events. Defaults as #9 (see programming note #18)

11) FORMULA = '&,&&' (DEFAULT = 'VL,BIT') Describes formula in input stream where & may be:

IC - intersecting cover mode
DC - disjoint cover mode
VL - dependent cover mode

where && may be:

BIT - each complex is a bit string
CHAR - this whole formula is a character string

$ $ $ $ $ $ $ $ $ Control Information $ $ $ $ $ $ $ $ $ $

12) COPY_QUANT = 'NO' (DEFAULT = 'YES') - indicates processed events should be written in SYSPRINT in addition to E_OUT_FILE

13) COPY_STAT = 'YES' (DEFAULT = 'NO') - indicates the class number of the event (given by CLASS_FORMAT or INDIV_SPEC) appears in the E_OUT_FILE (SYSPRINT also if COPY_QUANT = 'YES') after the last variable

14) BOOL_PAT = '0001'B (DEFAULT = '0100'B) - see the secondary parameter TRIM_VAR

15) INDIV_SPEC = 'YES' (DEFAULT = 'NO') - the program should expect the true class number to follow the last variable in each event (in the same format as the variables)

16) PERFORM = '&,&&,&&&' - indicates the processing to be done on the events

& may be - DISCRETE or QUANT - indicating whether the input is to be quantized or not. In either case trimming may be performed. One of these must always appear.

&& can be - EVAL - and is optional. It indicates that the events are to be evaluated against the formula (results are displayed)

&&& can be - STAT - and is optional. It indicates that the evaluation results should be compared to the true class and statistics kept

17) UNKNOWN = '   ' (DEFAULT = '*') - one to five characters which appear in place of an unknown variable value (FIELD_WIDTH is irrelevant)

18) PARTIAL = 'KEEP' (DEFAULT = 'SKIP') - indicates action upon encountering the unknown characters. If 'SKIP' is given the program never sees the event. If 'KEEP' is given, that variable value is assigned a value of -1.

19) TITLE = # - immediately following the primary parameters (or secondary if given). This number of complete lines is read in and printed at the top of the output (DEFAULT = 0)

20) TRIM_OPTIONS = 'YES' (DEFAULT = 'NO') - controls whether a second input request is issued for the secondary parameters.

* * * * * * * * *   Secondary Parameter Input   * * * * * * * * * *

These parameters are concerned with the control of data manipulation operations.

21) ORDER = '#,#,#,#...' - each # is one of the value class numbers. The events are reordered according to the order given above.

22) TRIM_VAR = '   'B - this is a bit string with one bit for each variable. Bit one corresponds to variable one and so on. If the bit is set to one (1), then the corresponding variable is to be kept, if the bit is set to zero (0) that variable is to be eliminated from all input events.

23) X(#) = '   'B - this represents a series of bit strings each of the given form. # indicates which variable the bit string applies to. The bit string is the length of

the cardinality of that variable (X(#)). Each bit represents a value the variable can take (beginning at zero (0) on the left). Unspecified variables are assigned 'DON'T CARE' bit strings. Each event is tested on the basis of these variable values and kept or deleted as a result.

a) If BOOL_PAT = '0100'B then a one (1) bit represents a value the event must have to be kept.

b) If BOOL_PAT = '0001'B then a one (1) bit represents a value the event must not have to be kept.

Multiple values for each variable are allowed. If more than one variable bit string is given, the event must satisfy all conditions before it is kept.

* * * * * * * * *   Title   * * * * * * * * * *

Immediately following the primary parameter input (or the secondary input if given), the program will read in 'TITLE' number of complete card images and reprint them verbatim at the top of the output listing. If TITLE = 0 then no input request is issued.

* * * * * * * * *   Quantization Thresholds   * * * * * * * * * *

If QUANT is specified in the PERFORM = ' ' parameter then the program expects to find a series of thresholds for each variable immediately following the title (if given) in the input stream. These are decimal optionally signed numbers.

This program converts unquantized (raw) data into quantized data given the necessary thresholds. The quantization is done as follows:

Suppose, for variable V, the thresholds are T1, T2, ..., TN.
Let X be a raw value for variable V.

```
If      X<=T1, then the quantized value is 0
If T1<X<=T2, then the quantized value is 1
If T2<X<=T3, then the quantized value is 2
                 .
                 .
                 .
If TN<X,       then the quantized value is N
```

Partial data (incomplete input vectors--some variable values missing) may be handled in one of two ways, as specified by the user:

1) Incomplete input vectors will be skipped--not quantized and not output.

2) Unknown variable values will be pseudo-quantized to -1 and output.

On input (if PERFORM = 'QUANT,...') the input stream is assumed to contain (# of levels -1) threshold values for each variable. The last thresholds are echo printed to verify that the proper number of thresholds were read.

* * * * * * * * *   Event Input   * * * * * * * * * *

The program will look in the E_IN_FILE for the appropriate number of events. Each event is a string of numbers (#_VAR long). If INDIV_SPEC = 'YES' then one additional number per event is read in and interpreted as the true class. Each variable must be contained within the FIELD_WIDTH specified in FORMAT_E_IN. Events may take more than one card image but only #_PER_LINE variables are read from each card. Two events must not share the same card. After #_VAR (or #_VAR+1) numbers are read in, the rest of the card is ignored.

* * * * * * *   Parameter and Formula Specification   * * * * * * * *

The program will look in the FORMULA_FILE for the formula and its associated parameters. If FORMULA = 'BIT,...' then list-directed input is used exclusively. If FORMULA = 'CHAR,...' then the parameters are read with list-directed input but the formula in edit-directed input.

The first number the program expects is the total number of complexes in the cover. Next it expects to find a pair of numbers for each class. The first of the pair gives the class number while the second gives the number of complexes in that class.

If FORMULA = 'BIT,...' then the total number of complexes is read in. Each complex is assumed to be a bit string (' 'B). The string is the length of the sum of the cardinalities for each variable that is kept. A one (1) bit indicates an acceptable value while a zero (0) bit indicates an unacceptable one.

If FORMULA = 'CHAR,...' then the formula is read in as characters (A format) and parsed to appropriate bit strings.

* * * * * * * * *   Output   * * * * * * * * * *

Output from the program is self-explanatory. Each phase of the program prints messages describing what it has done. In addition, for each dynamic allocation performed, a message is printed showing the remaining unused core in the program's region.

Immediately after the title, input parameters are echo printed so you can verify what the program used as parameters. If quantization was specified the last thresholds read are echo printed so you can check that events were not read as thresholds.

If E_OUT_FILE is specified then the quantized and processed events are written into E_OUT_FILE in the FORMAT_E_OUT given (if COPY_STAT = 'YES' then the true class is written after the last variable in the same format.

* * * * * * * * * Programming Notes: * * * * * * * * * * *

1) Data for two different events must not appear on the same line.

2) All cards in the event input that contain characters other than numeric data, blank, minus, period, plus or the unknown character and all blank cards are ignored. They are reproduced in the output stream but not written into the E_OUT_FILE. They must not contain data.

3) The specifications for '#_INEVENTS' and 'CLASS_FORMAT' are mutually exclusive.

4) All input data is checked for unspecified values. If you wish to eliminate variables specify PARTIAL = 'KEEP' so events are not eliminated on the basis of these variables.

5) Using the sort option involves substantial dynamic core allocation. Make sure you specify enough core.

6) The FRECORE subroutine will only work on IBM systems using MVT, MFT or PCP. Disable the function call for other systems.

7) Classes are assumed to be numbered consecutively from 0 to (#_CLASSES - 1). Variables are quantized consecutively from 0 to cardinality - 1.

8) All options apply to the events as input to this program not the trimmed or quantized events (except where noted).

9) On input, each variable for each event must occupy a field width that is uniform over the input set (as given by FORMAT_E_IN = 'FIELD_WIDTH'). The variable value may be anywhere in this field. NOTE: Blanks are not interpreted zero.

10) If MODE = 'IC' then the overall percentage may deviate a small amount from the true value since multiple assignments are not taken into account (the last assignment is used).

11) Event input and output assume 80 ~~Bit~~ BYTE card images.

12) Dynamic allocation is done separately for parsings, quantization, evaluation, statistics and sorting phases.

13) If COPY_STAT = 'YES' then room must exist on the output line (E_OUT_FILE) for the additional item. This condition is not checked for.

14) If the program is to be used iteratively, all variables and defaults are reset before successive passes are begun.

15) '#_CLASSES' always indicates the total number of classes in the experiment. If MODE = 'VL' then the program expects to find complexes for (#_CLASSES - 1) classes. Statistics are not kept for unassigned events regardless of mode.

16) Trimming options are performed in the following order:

    a) (UNKNOWN,PARTIAL) - eliminate events on basis of unspecified values

    b) (TRIM_VAR) - eliminate specified variables

    c) (ORDER) - sort classes

    d) (X,BOOL-PAT) - eliminate events on basis of variable values.

17) 'BOOL_PAT' is used as the third argument in the PL1 'BOOL' built-in function. The first argument is the concatenated values of the array 'X' (unspecified elements set to 'DON'T CARE'). The second argument is the bit string representation of the event.

18) If neither 'FORMAT_E_IN' nor 'FORMAT_E_OUT' appear in the input, the defaults apply to both. If only 'FORMAT_E_IN' appears then 'FORMAT_E_OUT' is assumed identical to 'FORMAT_E_IN'. If only 'FORMAT_E_OUT' appears then the defaults apply to 'FORMAT_E_IN'. If '#_PER_LINE' or 'FIELD_WIDTH' are unspecified in either formats they take the last assigned value (can be the default).

19) If PARTIAL = 'KEEP' the statistics may deviate from their expected values. This is because unspecified values will match anything and results are unpredicatable.

20) The bit strings (complexes) of the input formula must match in length, the bit strings representing events after variable elimination is performed. If the complex is short, no events will match any complex.

* *

An example of the program input follows. The three pages are actually one contiguous input stream (consisting of two distinct problems, the program cycles through both separately).

	PARAMETER	OPTIONAL?	DEFAULT
1)	BOOL_PAT	YES	'0100'B
2)	COPY_STAT	YES	'NO'
3)	COPY_QUANT	YES	'YES'
4)	CLASS_FORMAT	(see note 3)	
5)	E_IN_FILE	YES	'SYSIN'
6)	E_OUT_FILE	YES	
7)	FORMAT_E_IN	(see note 18)	
8)	FORMAT_E_OUT	(see note 18)	
9)	FORMULA	YES	'BIT,VL'
10)	FORMULA_FILE	YES	'SYSIN'
11)	INDIV_SPEC	YES	'NO'
12)	ORDER (SECONDARY)	YES	
13)	PERFORM	NO	
14)	PARTIAL	YES	'SKIP'
15)	TITLE	YES	0
16)	TRIM_OPTIONS	YES	'NO'
17)	TRIM_VAR (SECONDARY)	YES	
18)	UNKNOWN	YES	'*'
19)	VAR_CARD	NO	
20)	X (SECONDARY)	YES	
21)	#_INEVENTS	(see note 3)	
22)	#_VAR	NO	
23)	#_CLASSES	NO	

SAMPLE INPUT - PART #1

```
  PERFORM='EVAL'  CLASSIFORMAT='0 4 1 4  , 1 4 1 4 '
  FORMATIEJIN='FIELDJUIDTH=2,#IPERILINE=17'
  TITLE=3 , #IVAR=17 , #ICLASSES=2
  VARICARD='2 2 5 5 5 5 3 3 3 3 5 5 5 5 3 3 3';
  ***************************************************
  TEST EVAL & STAT WITH TABLE EXAMPLE
  ********************%%%%%%%%%%%%%%%%%%%%%%*************
 -0 1 1 2 3 4 1 1 1 1 1 2 3 4 2 1 2
  0 1 1 1 1 4 1 1 1 1 1 2 3 1 2 1 2
  0 1 1 1 1 4 1 1 1 1 1 2 3·4 0 1 1
  0 1 1 2 3 4 1 1 1 1 1 4 4 1 2 0 2

  0 1 1 2 3 4 1 1 1 1 1 4 4 2 2 1 2
  0 0 1 2 3 4 1 1 1 0 1 2 3 0 2 1 0
  1 1 1 1 1 4 2 1 1 2 1 2 3 1 2 1 1
  1 1 1 2 3 4 2 1 1 2 1 2 3 4 2 1 1

   2 1 2
'0111111111111111111111111111111111111111111111111111111111111111111'B
'1111111111111111111111111111111111111111111111111111110100111111111'B
```

SAMPLE INPUT - PART #2

```
   FORMATJEJIN='FIELDJWIDTH=2,#JPERJLINE=24'
  FORMULA='BIT,IC'            INDIVJSPEC='YES'
  PERFORM='QUANT,STAT,EVAL'       TITLE=3
  TRIMJOPTIONS='YES'
  VARJCARD='3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3'
  #JINEVENTS=39
  #JVAR=23
  PARTIAL='KEEP'
  #JCLASSES=3      ;
  TRIMJVAR='00000000001011101111111'B
  X(11)='010'B     ;
$$$   &&&    ###    %%%
SUPER TEST RUN  LUCK
$$$   &&&    ###    %%%
    40    60
    .5    1.5
    .5    1.5
    .5    1.5
    .5    1.5
    .5    1.5
    .5    1.5
    .5    1.5
    .5    1.5
    .5    1.5
    .5    1.5
    .5    1.5
    .5    1.5
    .5    1.5
    .5    1.5
    .5    1.5
    .5    1.5
    .5    1.5
    .5    1.5
    .5    1,5
    .5    1.5
    .5    1.5
    .5    1.5

     THIS IS CANCER.RECTUM
70 0 0 1 1 1 1 0 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1
81 1 0 2 1 1 1 0 2 2 1 1 1 0 2 2 1 1 0 0 1 2 2 2
   THIS IS AN INTERSPERSED COMMENT
68 0 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 2 1
66 1 0 1 1 1 0 1 2 2 1 0 0 0 2 0 1 0 0 1 1 1 1 1
64 1 1 2 1 1 1 0 1 1 1 1 1 1 2 1 1 1 1 1 1 2 2 1
59 0 0 1 1 1 1 0 1 2 1 2 1 2 0 2 2 0 0 0 2 2 2 2
```

SAMPLE INPUT - PART #3

```
59 1 1 0 1 1 1 1 0 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1
28 0 0 2 2 1 2 0 2 2 1 2 1 2 1 1 1 1 1 0 1 2 2 0
55 1 0 2 2 1 1 0 2 2 1 1 1 1 1 1 1 1 1 0 1 1 1 1
73 0 0 2 1 1 0 0 2 2 1 1 0 1 2 2 1 1 0 0 1 2 1 1
59 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
35 1 0 2 1 1 2 0 2 1 1 2 1 1 1 1 1 1 2 2 2 1 1 1
62 0 0 1 1 1 1 0 2 1 1 1 0 1 1 2 2 1 0 0 2 2 2 2
36 1 1 0 1 1 1 1 0 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
80 1 0 1 1 1 1 0 1 1 1 2 1 2 2 2 2 1 1 1 2 1 2 2
82 0 0 2 1 1 2 0 2 1 1 2 1 1 2 1 1 1 1 1 1 2 1 1
65 1 1 1 1 1 1 0 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 2
73 0 0 1 1 0 2 0 1 1 0 2 1 1 1 1 1 1 1 0 2 2 2 2
59 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
72 1 0 1 1 1 0 1 1 2 * 1 1 1 1 0 1 1 1 1 1 2 * 1
72 0 0 1 1 0 0 1 1 1 1 1 0 1 0 2 1 1 0 0 1 1 1 2
61 1 0 2 1 1 0 0 2 2 1 1 0 1 2 0 0 1 0 0 1 1 1 2
69 1 0 1 1 1 1 0 1 2 2 1 1 1 1 1 1 1 0 0 1 2 2 1
81 1 0 2 1 1 1 0 2 1 1 1 1 1 1 1 1 1 1 0 1 2 1 1
 4 1 0 1 1 1 0 1 1 1 1 1 0 1 1 2 1 0 0 0 1 1 1 1
51 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
63 1 0 2 1 1 1 0 2 1 1 2 1 2 2 2 2 1 1 0 1 2 2 1
74 0 0 0 1 1 0 1 1 1 1 1 1 1 2 1 1 1 0 1 1 1 1 1
55 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 2 2 2
57 1 1 2 1 1 2 0 1 1 1 2 1 1 1 0 1 2 1 1 1 1 2 2
63 0 0 1 1 1 2 0 1 1 1 2 1 1 1 1 1 1 1 0 1 2 0 1
65 1 1 0 1 1 1 0 0 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
48 * 0 0 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1
54 0 0 2 1 1 1 0 1 1 1 1 1 0 2 2 2 1 1 1 2 2 2 2
47 0 0 1 1 1 1 0 2 1 1 1 1 1 1 0 2 1 1 0 1 2 2 2
79 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
58 0 * 2 1 1 1 0 2 1 1 1 1 1 1 2 1 1 0 1 1 1 1 1
63 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 0

   3     1 1,  2 1,  0 1
   '111111111111111111111111111111010'B
   '111111111111111111111111111111001'B
   '111111111111111111111111111111100'B
```