ELLA

(Every-gal-and-guy's Latent Lattice Analyser)

Includes "Super-Closure" by Ed Haertel
and "MLLSA" by Clifford Clogg

ELLA is a program to facilitate the construction of algebraic structures for dichotomous data and the testing of such structures given a latent class formalization of the error structure. It is based on a program originally written by Ed Haertel.

THIS PROGRAM WAS ORIGINALLY WRITTEN FOR a 32-bit extended DOS; it has recently been recompiled for modern platforms which may lead to sub-optimal interface features. This compilation is a standard Fortran Graphics program, i.e. a QuickWin that supports only one window. This allows the graphics used by ELLA, but may also lead to an automatic change to fullscreen mode which will be sub-optimal. You may have to get the window operating properly yourself, by choosing the font as you would for the MS-DOS console. Use ALT-ENTER to switch from full screen to windowed. For some irritating reason, the Window opened is larger than a normal DOS window--in fact, it's exactly one line larger on my screen than the window display, so you often can't see the last line. Hence a bit of fiddling with the font is necessary. Note that neither the "PAUSE" command nor the passing-commands-to-DOS operate successfully (they open and close a window instantaneously). Just open a separate DOS window.

WE ARE PLANNING TO SWITCH OVER TO THIS VERSION AND FORGET ABOUT THE DOS VERSIONS, and so we encourage users to try this out and tell us what's not working right.

FOR ALL VERSIONS:

There are significant changes and additions to the IBM version of this program (compared to "Super-Closure"). This documentation first covers the major changes, and then discusses each command.

1A. MEMORY USAGE:

ELLA dynamically allocates memory for large arrays as needed. Given today’s machines, few users will find this a real problem, but in some cases, your Windows set-up may not have allocated ELLA enough memory to perform some operation. In that case, ELLA will notify you in what routine the problem occurred, and warn you if it has been skipped, or if the results should be suspect. You can use the VERSION command to display a rough estimate of the amount of memory demanded by the program. If this amount is clearly less than what your system has made available, fiddle with the heap space, or eliminate other resident programs. If this doesn’t work, you can save memory in ELLA by reducing the number of structures that ELLA can handle at any time. This is done through the DEfaults SWitch command. Generally, not much memory is saved by this method unless the reduction is drastic. You can use 'PAuse' and then 'Mem /c |more' to get an idea of memory usage from within ELLA (ELLA syntax allows '$Mem /c |More'--see below).

1B. SYNTAX:

To choose a command, one can enter as many letters in a command as one wishes, or as few as one wishes, so long as that command can be distinguished from others, since there are commands beginning with the same letter. For example, now that there is a 'Select' command, 'S' does

not work for 'Save.' You must at least enter 'SA' for save (as opposed to 'SE' for 'Select'). But 'N' still works for 'Next.'

* A convention therefore used in all documentation is to print the name of each command with upper-case for all necessary input and lower-case for unnecessary input. When you see 'SElect', you know that 'S' will not be recognized, though 'SE' will.

The commands are basically not case sensitive, but there is one exception. If the first letter in the command is lower-case, the second letter in a command that requires at least two letters must also be lower-case. So 'SE', 'Se', and 'se' all are read as 'Select', while 'sE' is not recognized. Note that the computer only looks for as many letters as it needs to recognize the command--other letters are ignored, so 'NOTE' is interpreted as 'N'='Next.'

For commands that also have a number, the number follows the command name, but THERE MUST BE AN INTERVENING SPACE. Thus, 'N 42', 'NE 42'and 'Next-a-diddly-doo 42' are equivalent, while 'n42' (which would have worked in Haertel's Super-Closure) is simply read as 'Next' with no number attached. Also, if one wants to write comments directly to the log file, any command in which the first character is '!' is treated as a comment and ignored. Commands beginning with '$' are interpreted as DOS commands and are passed on to the operating system (eliminating the need to PAuse, see below).

One command, PRecedence, can take anywhere from 0 to 3 arguments. If no argument is given, ELLA prints out the relations of precedence used to form the current structure. If the structure was not formed using precedence commands, the precedences consistent with the structure are printed. If one argument is given, ELLA interprets this as a structure reference, and prints out the precedence relations in that structure. But PRecedence can also be used to SET relations of precedence. In this case, two item arguments follow PRecedence: there must be spaces between all elements, and a comma separating arguments, as in 'PR 6, 1'. If spaces are forgotten, ELLA is likely to intepret it simply as 'PR'. One can also add a structure reference: in this case, it follows the item references, as in 'PR 6, 1, 3'--make item 6 precede item 1 in structure 3.

Furthermore, there are now commands that take another command as their argument. In this case, they are the Help, DEfaults and SHow defaults commands. If Help is given without an object command, a summary of all commands is printed. If either of the last two is specified without an argument, a menu appears containing only the "major" defaults; but if an argument is given, ELLA goes straight to the defaults for that command. (See below).

Finally, some commands allow the direct input of a pattern as an argument. If Add has a pattern such as 0011 as an argument, it adds that pattern. GIve also allows for the use of binary representations.

2a. SETTINGS:

Second, ELLA saves information as to your last work environment in a file called 'ELLA.STS', so you don't always have to specify the same changes to the defaults. If it finds this file, it will use the last defaults for listing, structure building, etc. If it does not, it will use its master defaults. It also remembers the name of the data

file last used, and will prompt you if you want to use the same data. If you use this data, ELLA will also ask you if you want to restore the structures that were last in memory (along with their titles and precedence information). If those structures were formed when the data set had been sorted, ELLA will sort the data before loading them.

The settings file is updated when one exits the program through the 'EXit' command, but is not updated when one Quits. SAving data structures does not affect the settings file. Note that if you want to save your work from one session without overwriting the settings file, you can SAve and Quit, then LOad the structures when you want to re-use them.

2b. DEFAULT STRUCTURE:

ELLA is oriented around a single working default structure at any time, in contrast to Haertel's Super-Closure. When you first start, it assumes you are working on data structure #1, and all commands that refer to a data structure are interpreted as referring to the current default unless otherwise specified. When you want to refer to all structures, such as to SAVE all or ERASE all, enter a '0' as the structure reference.

3. STRUCTURE BUILDING:

A) SELECT

There are four basic ways to put a structure together, the 'Select' command, and then adding patterns or adding precedence relations. With the 'Select' command, one enters a whole group of patterns on the basis of some cut-off criterion, which can either be 'absolute' (i.e. all patterns greater than some specified frequency) or 'relative' to the average frequency (i.e. all patterns greater than 3.5 times the average frequency). Closure will be maintained as if one had entered in the patterns one at a time. One can suppress the listing of each pattern as it is added, or view it (see section on defaults).

4. B) NEXT / ADD PATTERNS

The second way is through the 'Next' and 'Add' commands, which allow you to maintain closure of the resulting structure under union, under intersection, under both, or under neither. (This last option is useful for using ELLA to form MLLSA batch jobs. See below.) The closure method is set under 'Defaults.' The program's start-up default is closure under intersection. One can also add patterns without Next by entering a binary representation of the pattern.

5. C) NEXT / ADD PRECEDENCES

The third way also uses the 'Next' and 'Add' commands, but instead of adding patterns on the basis of frequency, it adds precedence relations, which RULE OUT patterns. It ranks all the N(N-1)/2 precedence relations just as the patterns can be sorted by frequency. This ranking is on the basis of the ratio of Martin's I (William, not me). For more information on this measure, see Appendix 2 to this document. When ELLA is operating in this mode (again, set under 'Defaults'), it will print out the precedence relation, the value of the I-ratio, and, if requested under 'Defaults', print out the two-by-two table for each relation. One can skip around the order of precedence ratios just as if they were frequencies. Instead of putting forth a list of generated patterns, it prints the excluded patterns (if any).

5A. D) PRECEDENCES

The fourth way is the inclusion of precedence relations ala carte through the PRecedence command (see [1]). In this case, the default (changeable, see below) is for ELLA to list the transitive precedence relations as they are created. This can be turned off if one is sure one is including DIRECT preceders.

5B. It is possible to combine the frequency-based and precedence-based methods, but makes sense only as follows: given a structure built through adding patterns, precedence relations can then be imposed, and the set will remained closed.

WARNING: Occasionally, the combining of methods can lead to errors. If a structure has been formed by a method that uses precedence relations, and a pattern is added by frequency, the closure process may add a pattern already in the list. This error is trapped for in ELLA, which keeps track of how you have been building structures, but it can still happen if you try to modify structures LOaded from a disk file under these conditions.

6. LISTING

One can either LIST out only the patterns in a current data structure, or list out all patterns in the data with a '*' by patterns included in the current structure. In this latter mode, one can go backwards and forwards in the list. Also, LIST can print out rounded frequencies or not. Finally, LIST can print out a summary of the number of all patterns and respondents currently included in the structure.

7. SETTING DEFAULTS:

As explained above, the DEfaults and SHow defaults commands refer to specific 'object' commands. The choice of the method of putting the structure together (Precedence vs. Frequencies) is found under Next, and closure rules under Add or Check. Changing to the precedence relation way of building the structure automatically changes the default listing method to the full listing of all patterns in the data, though the user can of course change it back. SHowing the 'SWitch' command produces a list of the number of patterns in each structure. The various options discussed above are controlled through the 'Defaults' command. When any type of 'DEfaults' is specified, the computer goes through all the relevant defaults. If one simply enters <CR> for any question, the default is left unchanged. If no object command is given, a menu appears which contains options for the most commonly used defaults. To get the "minor" defaults from the menu, one would have to use the "All" option.

To examine the defaults, one uses the 'SHow' (defaults) command. This has the same syntax of taking an object command, or, if no object is specified, of going to a menu of the major defaults.

8. STRUCTURE HANDLING

In ELLA you can build up more than one algebraic structure at a time--the default set up is for nine or ten. However, the number of structures is changeable through the DEfaults SWitch command. Decreasing the number of structures can be a way to save memory space needed for computations (though savings will tend to be slight). See section on memory handling. In all cases, the number of items is limited to 16. (There may be different limitations for running MLLSA--see below.)

9. DEFAULT FILENAMES

ELLA takes the filename entered for the LOG (if there is one), and uses it as the basis for forming a number of different files, unless the user decides otherwise. By default, it will make a <FILENAME>.LOG log file, a <FILENAME>.SAV save file for completed data structures, and a <FILENAME><STRUCTURE#>.INP file for MLLSA (see below).

10. MAPPING

MAp will warn if the structure has not been built under intersection-closure (which is likely to lead to error), but will still attempt an inversion. If the structure has not been built according to any closure rules, the 1- and 0-vectors will not be added. MAp now not only gives the schema for creating the lattice along with the inversion but, given graphics capability of the computer, graphs it. Press <CR> when done viewing to continue. The MAp routine is limited to 500 nodes, the graphic routine in ELLA to 200 nodes. The graphic routine may not always produce the pretties version of the lattice--the picture can be changed using a different program. MAp prompts whether to create a file that can be used in the network-graphics program KRACKPLOT. If one exports the picture to KRACKPLOT, available from ANALYTIC TECHNOLOGIES, it can be edited and printed as well. YOU CANNOT PRINT DIRECTLY FROM ELLA.

11. DATAFILE HANDLING

Data files in a number of different ASCII forms can be read--it is not necessary that the user know the format, though formatted input is still possible. The 'standard' type is considered to be sequential format, where first the pattern vector is stored in 1's and 0's, and then the frequency follows. It is only important that there be spaces between each value, including the 1's and 0's in the pattern representation. (A program to facilitate the data handling, CONCON, is included with ELLA, and it stores data in such fashion). Data that are produced by the SPSS CROSSTABS /WRITE=ALL command are directly readable in the form in which SPSS makes them. This is a convenient way of producing contingency table data from individual data.

Thus to create the data for ELLA from SPSS for 4 variables, A, B, C and D, originally scored (1=yes) (2=no), the syntax would be

```
RECODE A B C D (1=1) (2=0) (ELSE=SYSMIS).
FILE HANDLE OUT /NAME='C:\ELLA\MYDATA.DAT'
PROCEDURE OUTPUT  OUTFILE=OUT.
CROSSTABS VARIABLES=A B C D (1,2).
```

```
/TABLES=A by B by C by D
/WRITE=ALL.
```

Note that in this form, ELLA expects the variables to be scored 1,2 and not 0,1, so don't recode; just tell ELLA to read this in as SPSS crosstabs output. Alternatively, you can convert it to standard form using CONCON. Standard form looks something like this:

```
0 0 0 0 34
0 0 0 1 23
0 0 1 0 45
0 0 1 1 88
... etc.
```

See an SPSS manual for details. If the variables are not in integer form, be sure to use the file input format that allows for this possibility.

NEW: In some cases (such as using ELLA to produce lattices for other applications, such as network data), you will not actually wish to analyze data. In this case, you can simply enter $NONE when asked for the filename (all caps). A data set with all patterns frequency 2 will be set up; you can use the add command to create the desired structure.

12. WRITING CDAS MLLSA INPUT FILES:

With the 'Write' command, ELLA uses the current structure to make an input file for Scott Eliason's CDAS program, with its version of MLLSA, which it either puts in the root directory, or in a C:\CDAS directory if you want. It has a series of defaults which can be changed, and automatically sets up a CPROB matrix given a generic error term (such as .9--anywhere in the structure where a 1 is found, it replaces with .9, and anywhere where a 0 is found, it replaces with .1). Particular values can of course be changed. Write also computes LCPROB start values with different possible methods, including the proportion of observed data in any pattern, and will trap for errors such as probabilties that do not sum to one or are zero (it will automatically adjust for the latter case). You may also fix parameters by entering the relative constraints for each pattern. The executable MLLSA file by default has the base name of the LOG file and the structure number as its name, and an INP extension. Value labels and variable names are not done.

A default option determines whether ELLA makes one long file for your different models, or separate input files for each model. When you want to end a long file, use the CLose command. (For CDAS options syntax, see below.)

At the same time, WRITE will make a version of the data file that CDAS needs, and save it using the same base name as the original data file, but with a 'L' after this name, to indicate that the left-most variable is moving fastest. (If the data was originally read in this format, it will not make this additional file.)

13. ON-BOARD MLLSA:

ELLA also incorporates a version of MLLSA. Options are specified in the DEfaults MLlsa command, and use the same code as CDAS, namely:

Option 1 -- do independence model
Option 2 -- assign cells to latent classes

```
            NOT CURRENTLY SUPPORTED
      Option 3 -- give details of first and last iterations
      Option 4 -- print cell frequencies, expected values, standardized
            residuals, and Freeman-Tukey residuals
      Option 5 -- use previous final values as new start values:
            NOT CURRENTLY SUPPORTED
      Option 6 -- Check local identifiablity of parameters
      Option 7 -- Print matrix of derivatives
```

Other options: this version of MLLSA has two additional options which, for compatibility with the Write routine, are not entered on this options string, but separately through the DEfaults MLlsa command. The first is to have option 6 also lead to the computation of the variance/covariance matrix of parameters. The second is to "Beep" when the analysis is finished (useful for big data sets).

Results are printed to a special file. You can review the results with the REview command--output is printed to the screen and, by default (which can be overriden using DEfaults REview), to the log file. The temporary results file will then be over-ridden. More information is available from online help.

14. COMMANDS:

Below is a list of all commands. Each command is written as follows: the capitalized letters indicate the minimum letters that must be entered for the command to be recognized. A # indicates a number referring to a structure, while ## indicates a number referring to an item, and ### indicates a number referring to a pattern (or, in the case of Next building through precedences, to a precedence relation). B indicates the direct binary representation of a pattern (ex: 0110). [#] indicates that the number is optional. [ob] indicates an optional "object" command, namely another command as an argument. [str] indicates an optional string. / indicates a choice of argument.

```
                    LIST OF COMMANDS
Add [#/B]     : Add current pattern or precedence relation to data structure #.
              If B given, add pattern B to default structure.
Check [#]     : Check result of adding current pattern (not precedence
              relation) to data structure #
DEfaults [ob]: Change default settings for specified command.  If no command
              specified, go to menu with most likely default groups.
DUplicate [#]: Duplicate structure # to next available position.
ERase [#]     : Erase structure #.  If [#] is 0,
              all data structures are purged.
EXit          : Equivalent to SAve followed by Quit, plus settings file updated.
File          : Switch to a new data file without quitting the program.
Give ###/B    : List which patterns (if any) in the current structure
              produce pattern ## under current closure rules.
              Trivial intersections/unions such as those that include
              the pattern itself are ignored.
GUttman [#]   : Produces a Guttman scale for the items based on marginals
              and includes the precedence relations.
Help [ob]     : Print out information about object command.  If no command
              is specified, this list is printed.
Index [#]     : Lists INDEX NUMBERS of patterns in structure #.
LIst [#]      : Lists all patterns in structure # (see section 6).
```

```
LOad          : Loads previously stored structures from a .SAV file.
MAp [#]       : Generates lattice MAP of structure #.
MLlsa [#]     : Immediately analyze with Mllsa.
Next [###]    : Get Pattern/Precedence #-Next if No Number Specfied.
Open          : Open new file for data output.
PAuse         : Allows the user to enter in DOS commands, such
              as file manipulation, without having to quit ELLA.  A single
              command is executed and then ELLA resumes; if one wishes to
              enter more than one command, type 'COMMAND' first, as
              many DOS commands as wanted, and then 'EXIT'
              to return to ELLA.  **THIS COMMAND MAY BE DEFUNCT ON MODERN OS**
PRecedence [##, ##][,][#]
            : Displays relations of precedence among items used to
              create structure # if no item references given, otherwise
              adds a precedence relation from the first to the second item
              specified in the given (or default) structure.
RAnk          : Ranks the items by 'hardness' as if they were unidimensional.
REview        : Review MLLSA output.
Quit          : End immediately without saving.
SAve [#]      : Save structure #.  If [#] is 0,
               all data structures are saved.
SElect        : Choose a set of patterns by cut-off criteria
SHow [ob]     : Prints current default settings (see 'DEfaults', and
              main text).
SOrt          : Sort the data by frequency, so that one may begin with
              the most common response patterns.  This is useful since
              ELLA now reads data files in different forms.
SWitch #      : Changes the default structure to #.
Title [str]   : Gives a structure a mneumonic title, (the specified string)
              which is later used when SAving it or Writing.
Undo          : Undo last command.
Version       : Print version number and memory requirements.
Write [#]     : Makes an input batch file for MLLSA (see section 12 for
              details).
```

15. ORGANIZATION OF COMMANDS

The commands may be broken down into the following categories:

A. File handling and Manipulation:

- FIle
- Open
- SAve

B. Default control:

- DEfaults
- SHow

C. Information about data set:

- GIve
- RAnk
- SOrt

D. Structure building:

- Add
- Check
- GUttman
- Index
- LIst

MAp
Next
PRecedence
SElect
Undo

E. Structure handling:
DUplicate
ERase
LOad
SWitch
Title

F. Model fitting:
MLlsa
REview
WRite

G. Other ELLA:
EXit
Help
PAuse
Quit
Version

16. A TYPICAL SESSION

An example of a typical session is provided in the run package as 'SAMPLE.LOG', and may be viewed with an editor or printed. Comments are added using the ! (see paragraph 1).

1. A session begins with the loading of a datafile, either the last one used (ELLA prompt) or a new one. Note that the file-type must be correctly specified--ELLA does not trap incorrectly specified file types. If you see a lot of '1's and '0's for cell frequencies, there was probably a misspecification either of file type or number of variables. Also, note that the correct path must be specified if the file is not in the current (ELLA) directory.

2. The defaults are set for the type of building that is to be done. The data may be sorted by frequency. Information about the data set may be gotten before the structure is built.
Old structures can be loaded in, or only new ones used. NOTE: it does not matter if structures formed in an unsorted data file are loaded in after the data is sorted or vice-versa.

3. Structures are produced by
   a. Adding patterns through Next and Add;
   b. Adding patterns through SElect;
   c. Adding precedence relations through Next and Add;
   d. Adding precedence relations through PRecedences; or
   e. Forming a GUttman scale.

4. A model is fit using MLLSA. After the routine is finished, the results are REviewed.

5. A descriptive title is given to the structure, possibly including

specifics as to the MLLSA model specifications.

6. One SWitches to another structure slot and produces another structure, etc.

7. Structures may be saved for later contemplation.

8. The program is exited.

NOTE: The SHow SWitch command, which lists the current structures in memory, also gives the Raferty's BIC (Bayesian Inference Coefficient) for models that have been fit with MLLSA, so as facilitate the comparison of many models. For this reason, if you plan to fit more than one model of the same structure (such as a "strong" model, or different conditional probability restrictions), it makes sense to DUplicate the structure to another slot before running the next model, so as to be able to compare BICs--and it also helps to Title the structures in a way that the differences between MLLSA models are clear.

17. MORE HELP AND DEBUGGING.
ELLA has online help for each command, which will tell you in more detail about command syntax, arguments, and defaults. Other information is available on-line regarding latent class analysis from within the Write and MLlsa commands.

Bugs are bound to remain in the program, and when errors do occur, make sure to save the log file (and other associated files, such as 'SAV' or 'INP' files, or even the 'ELLA.OUT' file if the bug leads to interruption of the MLLSA routine before the results can be REviewed). There is also a slim possibility that the program will notify you of some programming problem which has been corrected, but should still be looked into. (You will see a message such as '*DEBUG: Routine sort list exhausted'.) In this case also save the log file.

APPENDIX 1: The original instructions of Ed Haertel's SUPER-CLOSURE

'SUPER'-CLOSURE: PROGRAM FOR INTERACTIVE MODELING OF
'SUPER'-CLOSURE: PROGRAM FOR INTERACTIVE MODELING OF
LATENT RESPONSE PATTERN LATTICE STRUCTURES.

THE BASIC IDEA OF THIS PROGRAM IS THAT A SET OF (TRUE OR LATENT) PERFORMANCE STATE PATTERNS REPRESENTS THE UNION OF ONE OR MORE DISTRIBUTIVE LATTICES REPRESENTING UNIQUE ABILITY CONFIGURATIONS ENABLING EXECUTION OF THE RESPECTIVE ITEMS. IF EACH ITEM CAN BE SOLVED USING EXACTLY ONE SET OF UNDERLYING ABILITIES, THEN A SINGLE 'ABILITY STRUCTURE' SHOULD SUFFICE. IF ONE OR MORE ITEMS CAN BE SOLVED USING ONE OR MORE DISTINCT SKILLS, THEN TWO OR MORE ABILITY STRUCTURES MAY BE REQUIRED, ACCOUNTING FOR (OVERLAPPING) SUBSETS OF THE LATENT ABILITY PATTERNS.

THIS INTERACTIVE PROGRAM INCORPORATES CODE FROM THE EARLIER (NONINTERACTIVE) 'CLOSURE' PROGRAM, HENCE 'SUPER-CLOSURE'. IT CAN MAINTAIN UP TO 9 DATA STRUCTURES SIMULTANEOUSLY, AND CAN HANDLE LATENT RESPONSE PATTERNS ON UP TO 30 ITEMS. UP TO 2050 LATENT RESPONSE PATTERNS (WITH ASSOCIATED FREQUENCIES) MAY BE INPUT.

INPUT IS A FILE CONTAINING A LIST OF 1-0 VECTORS AND ASSOCIATED FREQUENCIES. THE LIST IS TREATED AS ORDERED; USUALLY ORDER WILL BE DESCENDING FREQUENCY. THE LIST COULD BE EITHER OF MANIFEST RESPONSE PATTERNS OR OF ESTIMATED FREQUENCIES OF LATENT RESPONSE PATTERNS. LATTER IS THE USE I HAVE IN MIND AT THE PRESENT. SUCH A LIST IS GENERATED, FOR EXAMPLE, BY 'DISATT' PROGRAM, WHICH ACCEPTS VECTORS OF TRUE POSITIVE AND FALSE POSITIVE PROBABILITIES FOR A SET OF ITEMS, TOGETHER WITH MANIFEST FREQUENCIES, AND CALCULATES CORRESPONDING LATENT RESPONSE PATTERN FREQUENCIES.

THE USER OF THIS PROGRAM REQUESTS PATTERNS SEQUENTIALLY FROM THE INPUT LIST, ALTHOUGH THERE IS AN OPTION TO MOVE AROUND IN THE LIST. PATTERNS ARE REFERRED TO ONLY BY THEIR INPUT LIST SEQUENCE NUMBERS. IT WILL BE HELPFUL TO HAVE A PRINTOUT OF THE INPUT LIST AT HAND WHEN RUNNING THIS PROGRAM.

AFTER REQUESTING NUMBER OF ITEMS, INPUT FILE NAME, AND INPUT FORMAT, PROGRAM READS ENTIRE LIST INTO MEMORY AND REQUESTS A COMMAND. EACH COMMAND CONSISTS OF A LETTER (UPPER OR LOWER CASE) READ IN COLUMN 1, USUALLY FOLLOWED BY A NUMBER. ONE OR MORE SPACES BETWEEN THE LETTER AND THE NUMBER ARE OPTIONAL. THE MEANING OF THE NUMBER (IF ANY) DEPENDS ON THE PRECEDING LETTER:

N GET NEXT PATTERN. THIS WILL ALWAYS BE THE FIRST COMMAND, AND RESULTS IN DISPLAY OF "CURRENT PATTERN IS" LINE. AN ALTERNATIVE FORM OF THIS COMMAND IS N<#>. THIS GETS THE PATTERN AT LOCATION <#> IN THE INPUT LIST. SUBSEQUENT 'N' REQUESTS WILL CONTINUE FROM THERE. IT IS PERMISSIBLE TO GO BACKWARD OR FORWARD.

C<#> CHECK RESULT OF ADDING CURRENT PATTERN TO DATA STRUCTURE INDEXED BY <#>. ANSWER IS EITHER 'ALREADY IMPLIED' OR ELSE A LIST OF NEW PATTERNS IMPLIED. FOR EACH OF THESE NEW PATTERNS, IF IT IS IN THE ORIGINAL INPUT LIST, ITS SEQUENCE NUMBER AND ASSOCIATED FREQUENCY ARE GIVEN.

A<#> ADD CURRENT PATTERN TO DATA STRUCTURE INDEXED BY <#>. OUTPUT OF THIS COMMAND IS IDENTICAL TO THAT OF C<#>. IF IMMEDIATELY PRECEDING COMMAND WAS C<#>, THE COMMAND 'A' MAY BE GIVEN WITHOUT A NUMBER. IN THIS CASE, THE <#> FROM THE 'C' COMMAND IS IMPLIED, AND REDUNDANT OUTPUT IS SUPPRESSED.

R<#> ERASES DATA STRUCTURE INDEXED BY <#>. IF <#> IS OMITTED, ALL DATA STRUCTURES ARE PURGED.

D<#> DUPLICATES DATA STRUCTURE INDEXED BY <#>. THE COPY CREATED WILL BE ASSIGNED THE FIRST AVAILABLE INDEX NUMBER.

S TRIGGERS REQUEST FOR OUTPUT FILE NAME, THEN UPDATES, SORTS, AND OUTPUTS ALL CURRENT DATA STRUCTURES. UPDATING REFERS TO A COMPARISON OF EACH PATTERN LIST TO THE INPUT DATA LIST FROM THE FIRST THROUGH THE CURRENT PATTERN. FOR EACH MATCH, THE INTERNAL SEQUENCE NUMBER IS UPDATED SO THAT WHEN DATA ARE OUTPUT, EXTANT PATTERNS ARE NOT DENOTED AS

GENERATED PATTERNS. NOTE THAT WORK CAN BE SAVED AT WILL, AT INTERMEDIATE POINTS IN ANALYSIS. S<#> SAVES ONLY DATA STRUCTURE INDICATED BY <#>. AT TIME OF SAVE, OPTIONAL COMMENTS MAY BE ENTERED. THESE ARE SAVED WITH THE DATA STRUCTURE(S). NOTE THAT UPDATING CAN BE SUPPRESSED BY PRECEDING S COMMAND WITH "N 1"

O OPENS NEW FILE FOR DATA OUTPUT. ONLY ONE FILE CAN BE OPEN AT A TIME, AND ONCE CLOSED, A FILE CANNOT BE REOPENED.

Q QUITS IMMEDIATELY WITHOUT SAVING

E 'EXIT', EQUIVALENT TO S FOLLOWED BY Q

U UNDO LAST COMMAND (CANNOT UNDO 'S', 'O', 'Q', 'E', OR 'U' ITSELF; UNDO IS MEANINGLESS FOR 'C'.)

L LISTS ALL PATTERNS INCLUDED IN A GIVEN STRUCTURE

M GENERATES LATTICE MAP OF A GIVEN STRUCTURE

I LISTS INDEX NUMBERS OF ALL PATTERNS IN STRUCTURE

APPENDIX 2: Ranking of precedence relations.

In constructing a measure that would allow one to automate the search for precedence relations, there are three things one would want to take into account. The first, obviously, is how small the (A1=0, A2=1) cell is for the A1->A2 relation. A cell that approaches zero is compatible with a relation of precedence. But at the same time, one would also want to take into account the size of the (A1=1, A2=0) cell, so as not to mistake a symmetrical relationship of association for a relation of precedence. Finally, one would want to take into account the over-all N as a way of determining whether a small count was significantly small or might be due to simply having very few cases.

In looking for such a measure, I have turned to an asymmetric index derived by William Martin, that assumes causality that is necessary but not sufficient (or the other way around). This measure goes to 1 when the (A=0, B=1) cell is zero, and goes to 0 when the product of the off-diagonal cells is the same as the produce of the diagonal cells. The formula is

$$I_b = \frac{ad-bc}{(a+b)(b+d)}$$

where a,b,c, and d represent the values in the 2x2 table thusly:

```
                A2
              0  |  1
               +---+---+
            0 | a | b |
     A1     --+---+---+
            1 | c | d |
               +---+---+
```

One can intuitively see how it controls for the marginals (as opposed to a just looking at the b cell). At the same time, it does help indicate when the data are compatible with the relationship of precedence A1->A2. We can also see that

$$I_c = \frac{ad-bc}{(a+c)(c+d)}$$

is the corresponding measure for the relation A2->A1. I therefore take the ratio of the two measures, I(b)/I(c), which is

$$I_r = \frac{(a+c)(c+d)}{(a+b)(b+d)}$$

Now this measure still does not take into account the total N, but since this is constant over the subtables being compared, it is irrelevant in this case. We can see that it basically is adjusting the ratio c/b by amount in the diagonal cells, and seems to be a fitting way of ranking precendence relations. In ELLA, for any two items, if I(r) is less than one, the order of the items is reversed and the inverse of I(r) listed.