IMPLEMENTATION OF DAVE: AN EXPERT SYSTEM FOR THE ANALYSIS OF THE WECHSLER ADULT INTELLIGENCE SCALES AND RELATED INFORMATION

by

GLEN ROSS WHITE

B. S., Mathematics, Louisiana State University, 1972

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

Kansas State University
Manhattan, Kansas

1985

Approved by:

Roger T. Hartley

Major Professor

LD
2668
.K4
1985
W54
c.2

A11202 965003

Acknowledgements

I would like to thank my major professor, Dr. Rodger T. Hartley, for his encouragement and patience during the completion of this project. His enthusiasm for Artificial Intelligence, especially Expert Systems, has been unavoidably contagious. I would also like to thank the other members of my committee, Dr. Elizabeth Unger and Dr. Austin Melton, for their constructive recommendations and support. Mostly, I would like to sincerely thank these three professors, more than any others that I have had, for providing an environmemt in the classroom which was always challenging, stimulating, and worthwhile.

During this past year, my wife has been working elsewhere. Therefore, I want to thank my two sons, David and Christopher, for coping with me as a concurrent student, mother, and father.

ILLEGIBLE

THE FOLLOWING DOCUMENT (S) IS ILLEGIBLE DUE TO THE PRINTING ON THE ORIGINAL BEING CUT OFF

ILLEGIBLE

THIS BOOK CONTAINS NUMEROUS PAGE NUMBERS THAT ARE ILLEGIBLE

THIS IS AS RECEIVED FROM THE CUSTOMER

Table of Contents

CHAPTER PAGE

1. Introduction 1

2. System Capabilities 4

3. Rule Design 10

4. Differences Between Implementation and Original Design 14

5. Implementation 17

5. System Limits 19

6. Conclusions 22

Bibliography 24

APPENDIX

A. Sample Report 25

B. Sign-rules 33

C. Observation & Background Menus 75

D. SD-Qualifeirs 79

E. Clinical Entities 81

F. Menu Handler 82

G. Evaluate Sign-rules 90

H. Evaluate SD-Qualifiers 94

I. Change SD-Qualifiers 96

J. Observation & Background Menu Execution 103

K. Change Observation & Background Menus 106

L. Make File Changes Permanent 111

M. Change Clinical Entities & Local Scores 113

N. Produce Output Report 117

O. Change Sign-rule 122
P. List of Files 138
Q. Users Manual 139
R. Change Sign-rule Algorithm 155
S. Change SD-Qualifier Algorithm 159
T. Inference Engine Algorithm 161

CHAPTER 1

INTRODUCTION

This is a report on the implementation of Dave: an expert system for the analysis of the Wechsler Adult Intelligence Scales (WAIS) and related information. Dave was designed by Richard C. Pierce as his Master's Report at Kansas State University in 1984. Dave is named for David Wechsler, the Chief Clinical Psychologist at New York City's Bellevue Hospital who developed the Wechsler Adult Intelligence Scales (WAIS) as a measure of intelligence. It has been clearly demonstrated through extensive research that patterns of test scores on this test and other similar ones can be indicators of other psychological clinical entities. These include neurosis, psychosis, antisocial personality disorder, organic brain syndrome, mental retardation, and giftedness. Dave is a tool used by the clinical psychologist in analysing test scores, observations of the clinical psychologist during administration of the test, and background information to produce a document which would list the most probable clinical entities present and a description of the rules which indicate them. This document would be used by the clinical psychologist to focus in on problems during the initial therapy sessions.

The technique used in the design and implementation of Dave is a sub-area of Artificial Intelligence (AI), the rule-based expert system. It uses two types of rules, the sign-rules and the sd-qualifiers. Richard Pierce developed the sign-rules

from the works found in Ogdon (1977) and Gilbert (1978). There are currently 146 rules which match test score, observation, and background information patterns and indicate a list of possible hypotheses which are likely to be true if the pattern is present. Each entity also has a qualifier which indicates the certainty of the entity for this pattern. See the section on system design for a detailed explanation of their structure and use. A listing can also be found in Appendix B. The other type of rule is the sd-qualifier. It is a list of all of the major clinical entities and the patterns of data which would indicate that a particular clinical entity is definitely present (Superqualified), or that it is definitely not present (Disqualified). If that clinical entity is disqualified, it is not given further consideration. See the section on system design for detailed information on its structure and use. A listing is also contained in Appendix C.

Most expert systems are designed to interpret, predict, diagnose, debug, design, plan, monitor, repair, instruct or control something. This system does not truly diagnose an individual as having one of the previously stated clinical entities but it does interpret the results of the WAIS test along with observation and background information. LISP is the most common language in the AI field and is particularly appropriate for this project because the major task being performed is list manipulation. The rules themselves are lists which consist not only data but also of expressions which are evaluated to determine the actions to be taken with the data.

The rules, the major menus, the clinical entities, and some of the score names are lists which are assigned to an identifier such as sign-rules using the setq command. Each of these setq commands is contained in its own separate file. This allows the data within the rule to be changed and written permanently into the file by the clinical psychologist without knowing any LISP code (See system capabilities and Appendix L). Although there is great flexibility built into Dave for the user to expand, tune, and modify for local conditions, it does not learn on its own or measure its own performance. This capability is discussed in Politakis (1882) and Stefik, et. al. (1983).

CHAPTER 2

SYSTEM CAPABILITIES

Dave is a menu driven system which allows a clinical psychologist user to enter current and previous psychological test results, observations and background information and receive an evaluation of the data. The user has the capability to add or change the clinical entities being evaluated, incorporate the results of tests other than the WAIS when given routinely by him, add to or change the observation and background menus, and add to or change the rules being evaluated. This section will cover each of these capabilities in detail.

When Dave is loaded, the evaluation menu will appear. This menu is as follows:

EVALUATION MENU

1. Enter previous WAIS scores.
2. Enter current WAIS scores.
3. Enter local scores.
4. Enter observation & background information.
5. Perform evaluation of test results.
6. View latest report on terminal.
7. Print latest report.
8. Produce report with all rules true.
9. Teach Dave Menu.
10. Exit Dave.

The first two items allow the user to enter the results of a current or previous WAIS test into the computer. Dave prompts the user for the results of each subtest in turn and then calculates several composite scores. These scores are initialized to zero; therefore, if they are not recorded prior to evaluation of the data, they are assumed to be zero.

Often a series of tests are administered in addition to the WAIS. In this case, the local clinical psychologist may elect to have these test results incorporated into the rules being evaluated. Item 3 is essentially the same as the first two but it works on this list of scores which have been added locally (See Item 3 of Teach Dave Menu below). These scores are also initialized to zero.

Item 4 presents the user with ten menus which are lists of possible observations of behavior during administration of the test or known background information. The user should enter the item number of each of the statements which are true for the examinee followed by a "+" . The list of menu numbers and item numbers entered are added to a list called obs-list which is used in the interpretation. If this entry is not used before the evaluation is done, it will then be assumed that none of the menu items were true.

Item 5 is the interpretation of the data entered in Items 1 thru 4 using the sd-qualifiers and the sign-rules (See Appendix T, inference engine algorithm.) A report is produced and written to a file called "report". Item 6 will use the "more" facility to show the report on the screen. If a printed copy of the report is desired then Item 7 should be used.

During the production of the report a set of three evidence scores are determined for each clinical entity which has not been disqualified. They are, in order of importance,

observation, WAIS score, and background evidence scores. Care must be taken to consider these scores in relation to the maximum score possible for that clinical entity. For example, there may be twenty-seven possible rules which could cause a point for the WAIS score for psychosis but only five for mental retardation. If ten rules fire (i.e., are found to have their evidence present) for psychosis and all 5 fire for mental retardation then the score for psychosis will appear twice as strong even though every rule for mental retardation fired. For this reason Item 8 was included to provide a view of the scores for each clinical entity with every possible rule fired.

Item 10 is used to exit the system. If more than one person's test results are to be evaluated it is not necessary to exit the system to remove the previous person's results. Each time one of the first five items are executed it replaces the data which previously existed for that item. Care should be taken in the case where there is a previous WAIS test for the first person's evaluation and there is only one WAIS test for the second. The previous test scores will have to be entered manually with zero for each score.

The process of refining and updating Dave's menus and rules will fall primarily on the user. The user has the ability to evaluate them under actual conditions and compare Dave's findings with those of the acknowledged experts in the field. With the advantage of hindsight, they may evaluate past cases whose mental condition is precisely known and compare them with

Dave's evaluation of initial observations and test results. More than half of Dave's coding is devoted to its facility of allowing the clinical psychologist to improve Dave's menus and rules without calling in a LISP programmer for a rewrite. It also allows them to incorporate other test scores into the rules based upon further research or local policy. Item 9 is the Teach Dave Menu. This is used in order to make changes to the rules, the local scores, the clinical entities, or the observation and background menus. When Item 9 is executed the following menu will appear:

TEACH DAVE MENU

1. Change observation & background menus.
2. Change the clinical entities to be examined.
3. Change the local score names to be examined.
4. Change sign rules.
5. Change Superqualifier/Disqualifier rules.
6. Make file changes permanent.
7. Return to main evaluation menu.
8. Exit Dave.

Item 1 is provided in order for the user to add or change observation and background entries to the menus in Item 4 of the Evaluation Menu. (See Appendix R for algorithm).

Currently Dave provides a wide range of observations but only evaluates, scores, and produces a report for seven clinical entities. They are neurosis, psychosis, antisocial personality disorder, organic brain syndrome, mental retardation, and giftedness. If, in the process of evaluation of Dave, it is found that it would be desirable to add other entities on this same level such as depression, anxiety, or

schizophrenia, then they may be added using item 2. Clinical entities may also be deleted using this facility.

The desirability of incorporating the results of other tests into this evauation, either as part of the refinement process or to suit local conditions, has already been discussed. Item 3 is the means by which new score names are entered into the system. The user will be asked for a description of the score subtest and a one word abbreviation of the score subtest name. The program will perform a check to ensure that the abbreviated name does not conflict with other names used elsewhere. Rules governing the evaluation of these new data items should then be added to the sign-rules and/or sd-qualifiers using Items 4 and 5 below. The results of these new data items will be entered using item 3 of the Evaluation Menu.

Items 4 and 5 are used to change the rules which are evaluated by Dave to produce the report in Item 5 of the Evaluation Menu. If a new clinical entity is added, it is then likely that a sd-qualifier will be added for it, giving conditions for superqualification and disqualification. Also additional sign-rules or changes to existing sign-rules should be made for these new entities. Items 4 and 5 are the primary tools of the user in the refinement of Dave's set of rules. (See Apendices R and S for algorithms and Appendix T, the User's Manual for detailed description of its use.)

In Items 1 thru 5 of the Teach Dave Menu, the changes made

are only made in the current computer memory and not to the permanent file. This is desirable during the process of refinement because changes can be made without fear of damaging the permanent rules. Experiments can be carried out with different combinations of rules and/or clinical entities to see what the change in the results will be. After the user has produced a report which he is satisfied is better than the one he had under the permanent rules, he may choose to make the changes permanent using Item 6. He will be presented another menu asking which one of the sets of rules, scores, or clinical entities will be made permanent. Each of these rules, scores or clinical entities is a list which is maintained in a separate file. The process of making these lists permanent involves replacing the old file with one which reflects what is currently in memory. Warning: If changes are made using Items 1 thru 5 above, and item 6 is not used then the changes will be forgotten when LISP is exited or Dave is reloaded.

Items 7 and 8 are self explanatory.

There is a help facility which is available anytime there is a requirement for input from the keyboard. It is designed to present explanatory information related to the current function being performed and return to a state which is as close to the original state in which help was called as possible. To invoke the help facility just enter "help" in lower case letters.

CHAPTER 3

RULE DESIGN

There are two types of rules used by Dave, the sd-qualifiers and the sign-rules. There is a sd-qualifier rule for each clinical entity. Each rule has two parts. The first part checks a clinical entity for disqualification. If so, it is removed from further consideration. The second part checks for strong support of the clinical entity. If so, it is superqualified, an explanation of the supporting evidence is placed on its output report, and 100 points is added to the observation evidence score for that clinical entity.

The sign-rules are used to formulate possible hypotheses if the evidence contained in the rule is present. There are three types of evidence possible in the rule: WAIS score, observation, and background evidence. All evidence listed in the rule must be present for the rule to be fired, i.e. considered in the output report. The score evidence checks patterns of WAIS subtest results. The observation and the background evidence checks the presence of a menu and item number on the list of entries made by the clinical psychologist.

Each of the sign-rules may or may not be identified for the might-add list. This is information which would be of use to the clinical psychologist if that clinical entity is otherwise found to be present. It is identified by a key word "certain" and one of the main clinical entities. See Appendix T,

Inference Engine Algorithm for its use.

A sign-rule may have one or more hypotheses. Each hypothesis has three parts, the main, the secondary, and the additional. The main part of the hypothesis specifies an entity which may or may not be one of the main clinical entities. It indicates that the condition listed is present. To determine the level of certainty that that condition or entity is present, there is a qualifier clause. It is identified by the key word "qualifier" and a string such as "probably" or "sometimes may be latent".

The secondary part of the hypothesis gives a more detailed description or more specific hypothesis than the main. Its format is the same as the main except that if its entity is "none" then the qualifier is omitted. The secondary is not evaluated but it is provided as information on the output report.

The additional part of the hypothesis gives more evidence which the clinical psychologist may examine to provide additional support for the main and secondary hypothesis. Its format is the key word "additional" and a string description of the additional evidence. It is not evaluated but is provided as information on the output report.

The grammar for the rules are as follows:

⟨sign-rule⟩ ::= rule ⟨number⟩ ⟨evidence list⟩ ⟨hypothesis list⟩

⟨sd-qualifier⟩ ::= ⟨contained in clinical-entities⟩
disqualifier ⟨sdq-evidence⟩
superqualifier ⟨sdq-evidence⟩
⟨evidence description⟩

```
⟨sdq-evidence⟩ ::= ⟨evidence⟩ |
                   ⟨sdq-evidence⟩ ⟨logical operator⟩ ⟨sdq-evidence⟩

⟨evidence list⟩ ::= ⟨sign-rule evidence⟩ ⟨might-add flag⟩ |
                    ⟨sign-rule evidence⟩

⟨sign-rule evidence⟩ ::= ⟨evidence⟩ ⟨sign-rule evidence⟩ |
                         ⟨evidence⟩

⟨evidence⟩ ::= ⟨score⟩ | ⟨observation⟩ | ⟨background⟩ |
               ⟨absence of observation⟩ | ⟨absence of background⟩

⟨observation⟩ ::= observe ⟨menu number⟩ ⟨item number⟩

⟨background⟩ ::= bckgrnd ⟨menu number⟩ ⟨item number⟩

⟨menu number⟩ ⟨item number⟩ ::= ⟨contained in bckg-obs-menu⟩

⟨absence of observation⟩ ::= not ⟨observation⟩

⟨absence of background⟩ ::= not ⟨background⟩

⟨might-add flag⟩ ::= certain ⟨contained in clinical-entities⟩

⟨score⟩ ::= ⟨score check⟩ ⟨evidence explanation⟩

⟨score check⟩ ::= ⟨score comparison⟩ |
                  ( ⟨score check⟩ ⟨logical operator⟩ ⟨score check⟩

⟨score comparison⟩ ::= ⟨complex data⟩ ⟨relative operator⟩
                                          ⟨complex data⟩

⟨complex data⟩ ::= ⟨data item⟩ |
                   ⟨data item⟩ ⟨arithmetic operator⟩ ⟨complex data⟩

⟨data item⟩ ::= ⟨number⟩ |
                ⟨contained in all-score-list⟩ |
                ⟨contained in all-prev-score-list⟩ |
                ⟨contained in local-score-list⟩

⟨logical operator⟩ ::= and | or

⟨relative operator⟩ ::=     = | ⟨ | ⟩ | ⟨= | ⟩= | ⟨⟩

⟨arithmetic operator⟩ ::=    + | - | * | /

⟨evidence explanation⟩ ::= ⟨list of text strings⟩

⟨hypothesis list⟩ ::= ⟨hypothesis⟩ ⟨hypothesis list⟩ |
                      ⟨hypothesis⟩

⟨hypothesis⟩ ::= hypothesis ⟨main⟩ ⟨secondary⟩ ⟨additional⟩

⟨main⟩ ::=  main ⟨qualifier⟩ ⟨entity⟩
```

```
⟨secondary⟩ ::= secondary ⟨qualifier⟩ ⟨entity⟩ |
                secondary ⟨entity⟩

⟨additional⟩ ::= additional ⟨string⟩

⟨qualifier⟩ ::= qualifier ⟨string⟩

⟨entity⟩  ::=  ⟨contained in clinical-entities⟩  |
               ⟨one word string⟩
```

CHAPTER 4

DIFFERENCES BETWEEN IMPLEMENTATION AND ORIGINAL DESIGN

Dave was originally designed by Richard C. Pierce as his Masters Report at Kansas State University in 1984. During the implementation, his design was followed closely. However, the significant changes and enhancements should be pointed out for clarity.

The "test-retest" type of evidence for sign-rules was incorporated into the "score" evidence with no difference in the way it is treated. In the same area, the "sochist" evidence is assumed to be the same as "bckgrnd" therefore "sochist" is not used.

English language explanations were substituted on the final report for rule numbers and menu and item numbers. Because this produces a longer report the "does not support" category on the final report was deleted. If there is a need to obtain information regarding rules which were not used because the evidence was not present, Item 8 of the Evaluation Menu will produce a report with the output of all the rules present.

The inference engine algorithm has several major differences. Instead of forming separate lists for observation, WAIS score, and background rules, each rule is put onto the list used for the final report as the rule is evaluated. Although the three types of evidence scores are calculated, all of the output is maintained in one report for each clinical entity. These reports are presented in order of

highest score.

The original design of the inference engine called for special treatment for schizophrenia. Instead, schizophrenia was treated consistently as a secondary hypothesis of psychosis. The secondary hypothesis is designed to indicate a more specific diagnosis of the main hypothesis.

Several small changes were made for ease of use, or for simplification. The hypothesis number was dropped. The clinical entity specified in the main and secondary portion of the hypothesis is treated as an atom instead of a list. The menu item was changed from a letter to a number. None of these changes represent organizational or functional changes.

Enhancements were made to Dave as it was being built because they seemed to be logical extensions of the capabilities already called for in the design. Changing the rule number on the output report to an English language description of the rule was discussed above. The "help" feature was added as a convenience to the new user in areas where additional references might be needed to accomplish a task correctly. The learn module was mentioned in the design although no specifications were made. Therefore all of the rules, clinical entities, and observation and background menus were included in Dave's capability for user modification. Richard Pierce discussed the wide range of other tests which were routinely administered along with the WAIS, so it was a logical extension to include the ability for the user to

consider those tests results as Dave was being refined. All of these enhancements were made in the hopes that the system would be a valuable, flexible, and user-friendly tool for the clinical psychologist.

CHAPTER 5

IMPLEMENTATION

Dave is written in a particular dialect of LISP called Franz LISP which operates under the UNIX operating system (UNIX is a trademark of Bell Laboratories). The UNIX command "more" is used to display help screens and to display the permanent lists which contain clinical entities, local score names, both types of rules, and observation and background menus. In order to facilitate the translation of Dave into another version of LISP a list of all functions used is provided below. Franz Lisp uses the quotation mark (") or the vertical bar (|) to delimit strings. For an description of each of the following functions and key words see "The Franz LISP Manual" by J. K. Foderaro and K. L. Sklower.

Franz LISP Functions and Key Words

add, +
and

car, cdr
caseq
concat
cons

defun
Divide

eq, =, equal
exec

greaterp,),)=

implode

last
let
load

mapcar

add1
append

(also combinations i.e. caadar
close
cond

difference, -
do

eval
explode

lessp, (, (=
list

max

```
member                                  min
msg

neq, ()                                 nil
not                                     nth
null

or                                      outfile

plus                                    poport
pp-form                                 princ
print

quote, '                                ` (backquote used with comma)
quotient, /

read                                    readc
remove

set                                     setq
sort                                    sortcar
subl

t                                       terpr
times, *                                tyipeek

zerop
```

LISP requires 581.5K memory and Dave requires an additional 113.5K, for a total of 695K for normal operations of Dave. To run the Teach Dave Menu, additional memory is needed according to the type of changes desired. The additional memory is as follows: change Observation & Background Menus needs 16K, change clinical entities or local scores requires 1.5K, change sign-rules requires 29K, change sd-qualifiers requires 44K, and to make any file permanent requires an additional 1/2K.

CHAPTER 6

SYSTEM LIMITS

It should be understood that Dave is not a program which allows a computer to diagnose and treat persons with mental disorders. It is rather a tool used by a trained clinical psychologist to interpret test results according to a well accepted set of rules. The result will be a report which will assist the psychologist in planning therapy for the examinee. There are many reasons why Dave is not capable of making a firm diagnosis and several dangers inherent in depending too heavily upon the results provided by Dave.

The clinical psychologist administering the test must make a number of subjective judgements when deciding which one of the observation and background menu items are present. For example, one of the menu items reads "Frequently gave irrelevant details". The user will decide what constitutes frequently. This type of judgement or simple clerical error in failing to make an entry may cause a rule to be used incorrectly. Dave has no ability to detect these type errors nor does it have the ability to evaluate the rules themselves.

Although the data upon which the sign-rules are based has been developed through extensive research, it is possible that errors may have been made by Richard Pierce in the original drafting of these rules or in my translation of them into a slightly more strict format. Until these rules and the sd-qualifiers are tested extensively Dave will deserve little trust by the user as a worthwhile tool.

There has been a strong emphasis in the development of Dave upon flexibility and the ability of the user to modify it to fit his situation. In doing so standardization and the ability to verify the validity of changes has been sacrificed. Local psychologists may make rule changes or additions which may provide results which are not accepted scientifically by their colleagues.

The facilities for adding and changing rules is designed for use by an individual who knows nothing about LISP code. As nothing appears on the output report except the English language description of the rule, should the user makes a mistake in entering the part of the rule which is evaluated, then erroneous results may occur.

There are several weaknesses in the current rules which were discussed at length by Richard Pierce in his report. Dave has an inability to compensate for extremely high or low scores. Each subtest has a range of scores between 1 and 19. If an individual scores average in one area and very high in all other areas then Dave will interpret the average area as a very bad score. Further, there are no criteria for disqualifing giftedness. There are very few sign-rules which use background evidence. More sign-rules are needed which deal with giftedness and the case where IQ does not reflect the persons true abilities, such as when they are deprived of early intellectual stimulation. These areas should be considered by the psychologists in refinement of Dave's rules.

Little distinction is made between background and observation information. They are combined on the menus and considered by the system in exactly the same way. The only difference is the name of the function which is used in the sign-rule or sd-qualifier. If the rule specifies (observe 3 4) and another rule specifier (bckgrnd 3 4) then the same thing will happen except in the way that they will score the clinical entity involved in the rule. It is left to the person creating the rule to determine whether Menu Number 3, Item Number 4 is background information or an observation. No real harm to the system occurs if this is done properly but it is another place where human clerical error can permit flaws.

The first hypothesis of a rule which contains a major clinical entity will cause that clinical entity to receive points based upon observation, WAIS score, and background evidence. This score will determine the most probable clinical entity present in the examinee. There is no consideration for the maximun number or rules in the system for that clinical entity.

CHAPTER 7

CONCLUSIONS

The next step in the development of Dave is to measure its effectiveness in the areas of ease of use, the appropriateness of the output report, and the validity of the rules. It needs to be tested on actual cases to compare Dave's report with that of a qualified human expert. The facility for changing the rules which govern the system is an itegral part of the design, therefore refinements are expected to occur initially as well as throughout its use.

Flexibility was a major goal in the design and implementation of Dave. The ability of the clinical psychologist to add or delete clinical entities to be evaluated or incorporate other test scores in addition to the WAIS has given the user the flexibility to modify it to suit local conditions. He may even obtain a report which is slightly different for a particluar patient without altering the permanent files.

No system will be used if it is difficult to understand. Dave is menu driven. The user only needs to respond to queries from the screen or choose items from a menu to perform any function. The menus each contain explanatory information and the help facility will provide on-line explanations of all functions. No programming experience is required for the user to add or change rules even though they are actually creating new LISP code.

Saving the clinical-psychologist's time was another goal. To evaluate the test scores manually normally takes several hours for a trained clinical psychologist. It takes that same person three minutes to enter the scores and make entries from the Observation & Background Menus. The report will be available immediately for the psychologist to review and act upon while the patient is still in the office.

It is possible that the rules can be refined to the point that Dave would be more comprehensive than the normal clinical psychologist in the number of the rules which it considers. As more clinical psychologists look at the rules and evaluate their validity, the number of rules will grow to the point that it would be impractical for a clinical psychologist to manually evaluate every patients scores.

One of the major benefits of expert systems is that it encourages the technical users to more precisely define their science. A tool such as Dave will make it practical for the psychologist to consider one thousand rules where there may only be one hundred under study today. If this happens, then Dave will have played a major role in the advancement of both Computer Science and Clinical Psychology.

BIBLIOGRAPHY

Foferaro, J., Sklower, K., (1982). "The Franz LISP Manual" University of California, Berkley, California.

Gilbert, H. (1978). Interpreting psychological test data, volume I: test response antecedent, New York: Van Nostrand.

Ogdon, D. P. (1977). Psychodiagnostics and personality assessment : a handbook (2nd ed.), Los Angeles, California: Western Psychological Services.

Pierce, R. C. (1984). "DAVE: AN EXPERT SYSTEM FOR THE ANALYSIS OF THE WECHSLER ADULT INTELLIGENCE SCALES AND RELATED INFORMATION", Master's Report, Kansas State University.

Politakis, P. (1982). Using Empirical Analysis To Refine Expert System Knowledge Bases. Laboratory for Computer Science Research, Rutgers University.

Stefik, M., Aikins, J., Balzer, R., Benoit, J., Birnbaum, L., Hayes-Roth, F., & Sacerdoti, E. (1983). Building Expert Systems. Reading, Massachusetts: Addison-Wesley.

APPENDIX A, SAMPLE REPORT

****

DAVE'S EVALUATION OF POSSIBLE CLINICAL ENTITIES

****

********** antisocial-personality-disorder **********

The evidence score for examiner's observations was 103
The evidence score for WAIS test scores was 2
The evidence score for background information was 0

The supported rules for antisocial-personality-disorder
are as follows

superqualified
observe: Reality contact good
and observe: Lacks consideration for other
people's feelings and rights
and observe: History of violent behavior.

Main hypothesis antisocial-personality-disorder
secondary hypothesis narcissistic-character-disorder
score verbal IQ ⟨ performance IQ - 9

Main hypothesis tendencies for antisocial-personality-disorder
secondary hypothesis or tendencies for delinquency
score comprehension ⟨ verbal mean - 2

Main hypothesis antisocial-personality-disorder
secondary hypothesis
delinquency-or-scheming-without-regard-to-consequences
additional support? performance IQ ⟩ verbal IQ + 9
score picture arrangement ⟩ performance mean + 2

```
Main hypothesis  antisocial-personality-disorder
    observation Supercilious attitude

    observation No serious hearing impairment

    observation Clang associations on vocabulary


Main hypothesis    tendency for  antisocial-personality-disorder
    score picture arrangement ) performance mean + 2
    observation Flippancy


Main hypothesis  antisocial-personality-disorder
    observation A schemer

    observation Good visual-motor speed

    observation Good visual-motor coordination
```

********** psychosis **********

The evidence score for examiner's observations was 0
The evidence score for WAIS test scores was 5
The evidence score for background information was 0

The supported rules for psychosis are as follows

```
Main hypothesis    possible  psychosis
     secondary hypothesis    frequently tendencies for paranoid
                              schizophrenia-or-preschizophrenia
     score information ) verbal mean + 2

Main hypothesis    psychosis
    secondary hypothesis    other than paranoid  schizophrenia
     additional  support?    comprehension  ( digit  span  - 2,
                                  differentiates from organics
    score comprehension ( verbal mean - 2

Main hypothesis    possible  psychosis
    secondary hypothesis  schizophrenia-or-depression
```

score information) comprehension + 2

Main hypothesis possible psychosis
secondary hypothesis especially conditions for accute or anxious type schizophrenia
score object assembly (performance mean - 2

Main hypothesis possible setting of psychosis
secondary hypothesis conditions for schizophrenia-or-preschizophrenia
score block design) performance mean + 2

Main hypothesis possible psychosis
secondary hypothesis dissociative-or-schizoid- processes
score digit symbol (performance mean - 2

Main hypothesis psychosis
secondary hypothesis regressed schizophrenia
score digit symbol (performance mean - 2

Main hypothesis possible psychosis
secondary hypothesis depression
score digit symbol (performance mean - 2

Main hypothesis possible psychosis
secondary hypothesis possible tendency for schizoid-or-schizophrenia
score digit span)= arithmetic or digit span)= vocabulary

Main hypothesis psychosis
score vocabulary and information) comprehension + 2

Main hypothesis psychosis
secondary hypothesis schizophrenia
score block design) verbal mean + 2 and performence mean + 2

Main hypothesis psychosis
secondary hypothesis tendency for schizophrenia
score verbal and performance scatter) 5

The might add rules for psychosis are as follows

Main hypothesis possibly antisocial-personality-disorder
score performance IQ) 110

```
**************   organic-brain-syndrome   **********************

The evidence score for examiner's observations was 0
The evidence score for WAIS test scores was 3
The evidence score for background information was 0


The supported rules for organic-brain-syndrome are as follows


Main hypothesis   organic-brain-syndrome
     secondary hypothesis left-hemisphere-brain-damage
     score verbal IQ ( performance IQ - 9

Main hypothesis    possible  organic-brain-syndrome
     secondary hypothesis cerebellar-or-left-hemisphere-damage
     score comprehension ( verbal mean - 2

Main hypothesis    possible  organic-brain-syndrome
     secondary hypothesis    particularly to  right-hemisphere
     score object assembly ( performance mean - 2

Main hypothesis    contraindication of   organic-brain-syndrome
     score block design ) performance mean + 2

Main hypothesis    probability of   organic-brain-syndrome
     secondary hypothesis    particularly  right-hemisphere-or-
                                                    motor-area
     score digit symbol ( performance mean - 2
```

```
**********************   neurosis   **************************

The evidence score for examiner's observations was 0
The evidence score for WAIS test scores was 2
The evidence score for background information was 0
```

The supported rules for neurosis are as follows

```
Main hypothesis    possible  neurosis
     secondary  hypothesis   hysteria-or-cyclothymic-character-
                                                        disorder
     score verbal IQ ( performance IQ - 9

Main hypothesis    setting for  neurosis
     secondary hypothesis tendencies for obsessive-compulsive
     score information ) verbal mean + 2

Main hypothesis    obsessional features in  neurosis
     secondary hypothesis   impaired-judgement
     score information ) comprehension + 2

Main hypothesis   neurosis
     secondary hypothesis    possibly anxiety or hysteria  with
                             neurotic-or-inadequate-personality
     score digit symbol ( performance mean - 2
```

```
****************    other observations ************************
```

```
Main hypothesis   schizoid
  score block design ) verbal mean + 2 and performence mean + 2

Main hypothesis  psychotic-depressives
     score vocabulary and information ) comprehension + 2

Main hypothesis  obsessive
     score vocabulary and information ) comprehension + 2

Main hypothesis    possibly may still be  impairment-of-recent-
                                                        memory
     score current digit span )= previous digit span

Main   hypothesis   may   reflect   inflexibility-and-impaired-
                                                      judgement
     score comprehension ( verbal mean - 2

Main hypothesis  obsessive-doubting
     score digit symbol ( performance mean - 2
```

```
Main hypothesis    possible condition to be  epileptic
     score digit symbol ( performance mean - 2

Main hypothesis    possible conditions to be  beat-downtrodden-
                                                        homeless
     score digit symbol ( performance mean - 2

Main hypothesis    possible  hyperactive-or-manic-tendencies
     score digit symbol ( performance mean - 2

Main  hypothesis     probability  of   anxiety-frustration-and-
                                                        tension
     score digit symbol ( performance mean - 2

Main hypothesis    tendencies toward  depression
     secondary hypothesis    and tendencies toward  aboulia
     score digit symbol ( performance mean - 2

Main hypothesis setting for endogenous type  mental-retardation
     score block design ) performance mean + 2

Main hypothesis    possibly good  creative-ability
     score block design ) performance mean + 2

Main hypothesis     favorable-therapy-prognosis
     score block design ) performance mean + 2

Main  hypothesis     superior    visual-motor-coordination-and-
                                        perceptual-organization
     score block design ) performance mean + 2

Main hypothesis    possible  neurasthenia
     score object assembly ( performance mean - 2

Main hypothesis    tendencies to be  hyperactive
     score object assembly ( performance mean - 2

Main hypothesis    likelihood of tendencies for  depression
     secondary hypothesis    perhaps with  aboulia
     score object assembly ( performance mean - 2

Main hypothesis    likelihood of  anxiety-or-tension
     secondary  hypothesis    possible associated with  bodily-
                                    concern-or-castration-anxiety
     score object assembly ( performance mean - 2

Main hypothesis    possible tnedencies for  paranoid
     score picture arrangement ) performance mean + 2

Main hypothesis   favorable-prognosis-with-therapy
     score picture arrangement ) performance mean + 2

Main hypothesis     socially-adept-and-punctual
     secondary hypothesis    perhaps  normally-adjusting
     score picture arrangement ) performance mean + 2
```

```
Main hypothesis    conditions for  depression
     score comprehension ( verbal mean - 2

Main hypothesis    tendencies for  obsessive-compulsive
     score comprehension ( verbal mean - 2

Main    hypothesis      increased    probability     impulsive-
                      maladjusting-emotionally-unstable-behavior
     score comprehension ( verbal mean - 2

Main hypothesis     less-than-adequate-judgement
     secondary hypothesis    often  in-doubt-laden-persons
     score comprehension ( verbal mean - 2

Main hypothesis    tendencies for  intellectualizing
     secondary  hypothesis    possible  compensatory-reactions-
                               or-early-intellectual-hothousing
     score information ) verbal mean + 2

Main hypothesis    likely  poor-academic-achievement
     secondary hypothesis   poor-reading-ability
     score verbal IQ ( performance IQ - 9

Main  hypothesis    mental retardation setting   negate-acting-
                                                 out-tendencies
     secondary  hypothesis    especially   high-grade-familial-
                                                        type
     score verbal IQ ( performance IQ - 9

Main hypothesis     adolescents-delinquents-and-sociopaths
     secondary   hypothesis     although  may  not   hold   for
                             delinquents-with-good-reading-ability
     score verbal IQ ( performance IQ - 9



* * * * * * * * * * * * WAIS SCORES * * * * * * * * * * * * *

i = 13      information
s = 12      similarities
a = 8      arithmetic
v = 10      vocabulary
cm = 7      comprehension
ds = 10       digit span
pc = 13       picture completion
pa = 16       picture arrangement
bd = 16       block design
oa = 10       object assembly
cd = 10       digit symbol
viq = 100       verbal IQ
piq = 115       performance IQ
fsiq = 108       full scale IQ
df = 14       digits forward
db = 12       digits backward
```

* * * * * * * * * OBSERVATIONS AND BACKGROUND INFORMATION * * *

2 9 Clang associations on vocabulary
3 2 Flippancy
4 1 Good visual-motor coordination
5 13 Reality contact good
5 9 A schemer
5 8 Good visual-motor speed
5 6 Supercilious attitude
5 5 No serious hearing impairment
6 12 Overt hostility toward the examiner
6 10 History of violent behavior
6 9 Lacks consideration for other people's feelings and rights
7 1 Challenged examiner's authority, at least half-seriously

```
(setq sign-rules
      '((rule 1
              (score (greaterp viq (add piq 9))
                     (! verbal IQ ) performance IQ + 9!))
              (hypothesis  (main  (qualifier ! probably  most!)
neurosis)
                           (secondary (qualifier ! especially!)
                                      obsessive-compulsive)
                           (additional  ! also  anxiety  tension
neurasthenia!))
              (hypothesis   (main   (qualifier   !   probably!)
psychosis)
                           (secondary        (qualifier        !
particularly!) schizophrenia)
                           (additional ! none!))
              (hypothesis  (main  (qualifier ! none!)  organic-
brain-syndrome)
                           (secondary        (qualifier        !
particularly!)
                                      organic-and-asphasic-
conditions)
                           (additional
                            !    possible    right    hemisphere
dysfunctining!))
              (hypothesis (main (qualifier ! none!) depression)
                           (secondary none)
                           (additional ! none!))
              (hypothesis  (main  (qualifier  !  none!)  older-
normal-individuals)
                           (secondary none)
                           (additional ! none!))
              (hypothesis (main (qualifier ! none!)
                                patients-tending-to-remain-in-
therapy)
                           (secondary none)
                           (additional ! none!))
              (hypothesis    (main    (qualifier    !    none!)
overachieving)
                           (secondary   (qualifier   !    good!)
academic-achievement)
                           (additional ! none!))
              (hypothesis  (main (qualifier ! none!) psychotic-
depressive-trend)
                           (secondary none)
                           (additional ! none!)))
        (rule 2
              (score (lessp viq (difference piq 9))
                     (! verbal IQ ( performance IQ - 9!))
              (hypothesis (main (qualifier ! none!)
                                antisocial-personality-
disorder)
                           (secondary (qualifier ! none!)
```

```
                                        narcissistic-character-
disorder)
                              (additional ! none!))
                 (hypothesis (main (qualifier ! none!)
                                   adolescents-delinquents-and-
sociopaths)
                              (secondary (qualifier ! although  may
not hold for!)
                                        delinquents-with-good-
reading-ability)
                              (additional ! none!))
                 (hypothesis  (main  (qualifier ! none!)  organic-
brain-syndrome)
                              (secondary (qualifier ! none!)
                                        left-hemisphere-brain-
damage)
                              (additional ! none!))
                 (hypothesis (main (qualifier ! mental retardation
setting!)
                                   negate-acting-out-tendencies)
                              (secondary (qualifier ! especially!)
                                        high-grade-familial-type)
                              (additional ! none!))
                 (hypothesis  (main  (qualifier !  likely!)  poor-
academic-achievement)
                              (secondary (qualifier ! none!)  poor-
reading-ability)
                              (additional ! none!))
                 (hypothesis   (main   (qualifier   !   possible!)
neurosis)
                              (secondary (qualifier ! none!)
                                        hysteria-or-cyclothymic-
character-disorder)
                              (additional ! none!)))
          (rule 3
                 (score (and (lessp viq 81)
                              (and  (lessp  piq 111) (greaterp  piq
89)))
                        (! 89 ( performance IQ ( 111 and verbal IQ
( 81!))
                 (hypothesis  (main (qualifier ! none!)  learning-
disabilities)
                              (secondary none)
                              (additional ! none!))
                 (hypothesis    (main    (qualifier    !    none!)
delinquency)
                              (secondary none)
                              (additional ! none!)))
          (rule 4
                 (score (lessp i (difference vmean 2))
                        (! information ( verbal mean - 2!))
                 (hypothesis (main (qualifier ! none!)
                                   impoverished-early-environment)
                              (secondary (qualifier ! including!)
                                        lack-of-formal-schooling)
```

```
                    (additional ! none!))
          (hypothesis (main (qualifier ! none!) repressive-
defenses)
                    (secondary none)
                    (additional ! none!))
          (hypothesis  (main (qualifier ! tendencies  for!)
withdrawal)
                    (secondary none)
                    (additional ! none!))
          (hypothesis   (main  (qualifier  !  none!)   low-
scholastic-aptitude)
                    (secondary none)
                    (additional ! none!))
          (hypothesis   (main   (qualifier   !   possible!)
anxiety-or-hysteria)
                    (secondary none)
                    (additional ! none!))
          (hypothesis (main (qualifier ! tendencies for!)
                         acting-out-or-delinquency)
                    (secondary none)
                    (additional ! none!))
          (hypothesis  (main  (qualifier  !  setting  for!)
psychosis)
                    (secondary   (qualifier   !    none!)
schizophrenia)
                    (additional ! none!))
          (hypothesis  (main  (qualifier ! organic  setting
for!)
                         organic-brain-syndrome)
                    (secondary (qualifier ! none!)  left-
hemisphere-damage)
                    (additional ! none!))
          (hypothesis (main (qualifier !
possible!) organicity)
                    (secondary none)
                    (additional ! none!)))
     (rule 5
          (score (greaterp i (add vmean 2))
                 (! information ) verbal mean + 2!))
          (hypothesis (main (qualifier ! tendencies for!)
                         intellectualizing)
                    (secondary (qualifier ! possible!)
                              compensatory-reactions-or-
early-intellectual-hothousing)
                    (additional ! none!))
          (hypothesis  (main  (qualifier  !  setting  for!)
neurosis)
                    (secondary  (qualifier  !  tendencies
for!)
                              obsessive-compulsive)
                    (additional ! none!))
          (hypothesis   (main   (qualifier   !   possible!)
psychosis)
                    (secondary (qualifier
                              ! frequently tendencies
```

```
for paranoid!)
                                        schizophrenia-or-
preschizophrenia)
                        (additional ! none!)))
        (rule 6
              (score (lessp cm (difference vmean 2))
                     (! comprehension ( verbal mean - 2!))
              (hypothesis (main (qualifier ! none!)
                                  less-than-adequate-judgement)
                          (secondary (qualifier ! often!)
                                  in-doubt-laden-persons)
                          (additional ! none!))
              (hypothesis    (main   (qualifier   !   increased
probability!)
                                  impulsive-maladjusting-
emotionally-unstable-behavior)
                          (secondary none)
                          (additional ! none!))
              (hypothesis (main (qualifier ! tendencies for!)
                                  obsessive-compulsive)
                          (secondary none)
                          (additional ! none!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary  (qualifier !  other  than
paranoid!)
                                  schizophrenia)
                          (additional
                           ! comprehension ( digit span - 2,
differentiates from organics!))
              (hypothesis (main (qualifier ! tendencies for!)
                                  antisocial-personality-
disorder)
                          (secondary (qualifier ! or tendencies
for!)
                                  delinquency)
                          (additional ! none!))
              (hypothesis  (main (qualifier ! conditions  for!)
depression)
                          (secondary none)
                          (additional ! none!))
              (hypothesis   (main   (qualifier   !   possible!)
organic-brain-syndrome)
                          (secondary (qualifier ! none!)
                                  cerebellar-or-left-
hemisphere-damage)
                          (additional ! none!)))
        (rule 7
              (score (greaterp cm (add vmean 2))
                     (! comprehension ) verbal mean + 2!))
              (hypothesis (main (qualifier ! adequate to better
than average!)
                                  judgement-or-common-sense-or-
social-competence)
                          (secondary none)
                          (additional ! none!))
```

```
                (hypothesis (main (qualifier
                                    ! adequate to better than
average!)
                                  ability-to-delay-inpulsive-
reaction)
                            (secondary none)
                            (additional ! none!))
                (hypothesis  (main (qualifier ! possible setting
for!) psychosis)
                            (secondary  (qualifier  !  paranoid!)
schizophrenia)
                            (additional ! none!)))
          (rule 8
                (score (lessp i (difference cm 2))
                       (! information ( comprehension - 2!))
                (hypothesis (main (qualifier ! possible condition
of!) hysteria)
                            (secondary   (qualifier  !  referable
to!) depression)
                            (additional ! none!))
                (hypothesis (main (qualifier
                                    ! possible cronic involutional
paranoid!)
                                  psychosis)
                            (secondary none)
                            (additional ! none!))
                (hypothesis (main (qualifier ! none!) hysteria)
                            (secondary (qualifier !
none!) repression)
                            (additional ! none!)))
          (rule 9
                (score (greaterp i (add cm 2))
                       (! information ) comprehension + 2!))
                (hypothesis   (main  (qualifier   !   obsessional
features in!) neurosis)
                            (secondary    (qualifier    !    none!)
impaired-judgement)
                            (additional ! none!))
                (hypothesis   (main   (qualifier   !   possible!)
psychosis)
                            (secondary (qualifier ! none!)
                                       schizophrenia-or-
depression)
                            (additional ! none!)))
          (rule 10
                (score (lessp ds (difference vmean 2))
                       (! digit span ( verbal mean - 2!))
                (hypothesis (main (qualifier
                                    ! likelihood of clinically
significant!)
                                  anxiety-and-or-tension)
                            (secondary none)
                            (additional ! none!))
                (hypothesis    (main    (qualifier    !    none!)
distractibility)
```

```
                        (secondary none)
                        (additional ¦ none¦))
            (hypothesis   (main   (qualifier   ¦   possible¦)
organic-brain-syndrome)
                        (secondary (qualifier ¦ perhaps¦)
                                   left-hemisphere-damage)
                        (additional
                         ¦ digit span ‹ comprehension - 2 and
vocabulary - 2¦))
            (hypothesis    (main   (qualifier   ¦    possible
conditions for¦)
                              manic-depressive)
                        (secondary none)
                        (additional ¦ none¦))
            (hypothesis    (main   (qualifier   ¦    possible
conditions for ¦)
                              epileptic)
                        (secondary none)
                        (additional ¦ none¦))
            (hypothesis    (main   (qualifier   ¦    possible
conditions for¦)
                              psychosomatic-migraine)
                        (secondary none)
                        (additional ¦ none¦))
            (hypothesis (main (qualifier ¦ none¦) hysteria)
                        (secondary none)
                        (additional ¦ none¦)))
      (rule 11
            (score (greaterp ds (add vmean 2))
                   (¦ digit span › verbal mean + 2¦))
            (hypothesis (main (qualifier ¦ none¦)
                              unusually-good-attentive-
processes-or-low-anxiety)
                        (secondary (qualifier ¦ or¦)
                                   low-distractibility-or-
good-contact-with-environment)
                        (additional ¦ none¦))
            (hypothesis   (main   (qualifier   ¦   possible¦)
psychosis)
                        (secondary   (qualifier   ¦   perhaps
simple type¦)
                                   schizophrenia)
                        (additional ¦ digit span › vocabulary
+ 2¦))
            (hypothesis  (main (qualifier ¦ tendencies  for¦)
psychopathic)
                        (secondary none)
                        (additional ¦ none¦))
            (hypothesis    (main   (qualifier   ¦    possible
tendencies for¦)
                              obsessive-compulsive)
                        (secondary none)
                        (additional ¦ none¦)))
      (rule 12
            (score (greaterp df (add db 2))
```

```
                    (! digits forward ) digits backward + 2!))
              (hypothesis   (main   (qualifier   !   possible!)
organic-brain-syndrome)
                          (secondary (qualifier ! none!) brain-
damage)
                          (additional ! none!))
              (hypothesis (main (qualifier ! possible!)
                                excessive-rigidity-and-
concrete-thinking)
                          (secondary   (qualifier   !   in   the
extreme!)
                                     fragmented-thinking-under-
pressure)
                          (additional ! none!))
              (hypothesis (main (qualifier ! none!)
                                better-menory-span-than-memory-
for-symbol-patterns)
                          (secondary none)
                          (additional ! none!))
              (hypothesis  (main  (qualifier ! in  the  extreme
case of!) psychosis)
                          (secondary (qualifier ! none!)
                                     depression-or-
schizophrenia)
                          (additional ! none!))
              (hypothesis  (main (qualifier  !  schizophrenia!)
psychosis)
                          (secondary  (qualifier  !  paraniod!)
especially)
                          (additional ! none!)))
        (rule 13
              (score (or (lessp df db) (eq df db))
                     (! ^digits forward (= digits backward!))
              (hypothesis   (main   (qualifier   !   psychosis!)
psychosis)
                          (secondary (qualifier ! adult!)
                                     schizoid-or-schizophrenia)
                          (additional ! none!))
              (hypothesis (main (qualifier ! possible!)
                                blandness-or-negativism)
                          (secondary none)
                          (additional ! none!)))
        (rule 14
              (score (lessp a (difference vmean 2))
                     (! arithmetic ( verbal mean - 2!))
              (hypothesis (main (qualifier ! none!)
                                poor-ability-to-concentrate-
and-high-distractibility)
                          (secondary none)
                          (additional ! none!))
              (hypothesis (main (qualifier ! none!)
                                poor-arithmetic-skills-and-
reasoning)
                          (secondary none)
                          (additional ! none!))
```

```
            (hypothesis (main (qualifier ! none!) anxiety)
                        (secondary none)
                        (additional ! none!))
            (hypothesis  (main  (qualifier  !  none!)  low-
scholastic-aptitude)
                        (secondary (qualifier ! perhaps!)
                                   reflecting-rebellion-
against-authority)
                        (additional  ! information  ( verbal
mean - 2!))
            (hypothesis (main (qualifier ! possible!)
                              adolescent-delinquency-or-
sociopathy)
                        (secondary none)
                        (additional ! none!))
            (hypothesis    (main   (qualifier   !    possible
conditions for!)
                              psychosis)
                        (secondary   (qualifier  !  perhaps!)
schizophrenia)
                        (additional
                         ! comprehension ( verbal mean - 2 or
pict. arrangement ( performance mean - 2!))
            (hypothesis   (main   (qualifier   !   possible!)
organic-brain-syndrome)
                        (secondary  (qualifier  !   perhaps!)
left-parietal-area)
                        (additional ! none!))
            (hypothesis   (main   (qualifier   !   possible!)
alcoholism)
                        (secondary none)
                        (additional ! none!))
            (hypothesis    (main   (qualifier   !    possible
conditions for!)
                              hysteria-and-narcissisism)
                        (secondary none)
                        (additional ! none!))
            (hypothesis   (main   (qualifier   !   possible!)
suicide-potential)
                        (secondary none)
                        (additional ! none!)))
      (rule 15
            (score (greaterp a (add vmean 2))
                   (! arithmetic ) verbal mean + 2!))
            (hypothesis  (main  (qualifier ! likelihood  of!)
normal-adjusting)
                        (secondary none)
                        (additional ! none!))
            (hypothesis (main (qualifier ! none!)
                              low-distractibility-or-good-
concentration)
                        (secondary none)
                        (additional ! none!))
            (hypothesis    (main   (qualifier   !    possible
tendencies for!)
```

```
                              intellectualizing)
                        (secondary   (qualifier   !   as   in!)
obsessive-conditions)
                        (additional ! none!))
              (hypothesis   (main   (qualifier   !   possible!)
paranoid-overalertness)
                        (secondary none)
                        (additional ! none!)))
        (rule 16
              (score  (greaterp a (add ds 2)) (!  arithmetic  )
digit span + 2!))
              (hypothesis (main (qualifier ! none!) anxiety)
                        (secondary (qualifier ! of the kind!)
                              found-in-neurosis-
depression-or-preschizophrenia)
                        (additional ! none!)))
        (rule 17
              (score (lessp a (difference ds 2))
                     (! arithmetic ( digit span - 2!))
              (hypothesis (main (qualifier ! none!)
                              relief-from-anxiety-or-freedom-
from-distractibility)
                        (secondary   (qualifier   !   may   be
achieved through!)
                              fantasy-or-withdrawal)
                        (additional ! none!))
              (hypothesis  (main (qualifier ! possible  anxious
unreflective!)
                              hysteria)
                        (secondary none)
                        (additional ! none!))
              (hypothesis   (main   (qualifier   !   possible!)
organic-brain-syndrome)
                        (secondary  (qualifier  !  conditions
for!)
                              brain-damage)
                        (additional ! none!)))
        (rule 18
              (score (lessp s (difference vmean 2))
                     (! similarities ( verbal mean - 2!))
              (hypothesis  (main  (qualifier  !  inabliity  or
impaired ability for!)
                              thinking-abstractily)
                        (secondary none)
                        (additional ! none!))
              (hypothesis   (main   (qualifier   !   possible!)
psychosis)
                        (secondary (qualifier
                                    !   possible   especially
cronic paranoid!)
                              schizophrenia)
                        (additional ! none!))
              (hypothesis   (main   (qualifier   !   possible!)
organic-brain-syndrome)
                        (secondary (qualifier ! or conditions
```

```
for!) epileptic)
                        (additional ! none!))
            (hypothesis (main (qualifier ! none!)
                              antisocial-personality-
disorder)
                        (secondary none)
                        (additional
                         ! performance IQ ) verbal IQ + 9 and
pict. arrangement ) performance mean + 2!))
            (hypothesis (main (qualifier ! possible conditons
for!)
                              depression)
                        (secondary none)
                        (additional ! none!))
            (hypothesis  (main  (qualifier  !  setting  for
endogenous type!)
                              mental-retardation)
                        (secondary none)
                        (additional ! none!))
            (hypothesis (main (qualifier ! conditions for!)
                              beat-homeless-downtrodden)
                        (secondary none)
                        (additional ! none!))
            (hypothesis  (main (qualifier ! none!)  cultural-
deprivation)
                        (secondary none)
                        (additional ! none!)))
      (rule 19
            (score (greaterp s (add vmean 2))
                   (! similarities ) verbal mean + 2!))
            (hypothesis    (main    (qualifier    !    none!)
intellectualizing-defenses)
                        (secondary none)
                        (additional ! none!))
            (hypothesis    (main   (qualifier   !   favorable
prognosis for!)
                              continuing-in-therapy)
                        (secondary  (qualifier !  none!)  no-
early-termination)
                        (additional ! none!))
            (hypothesis    (main   (qualifier   !    possible
conditions for!)
                              obsessive-compulsive)
                        (secondary none)
                        (additional ! none!))
            (hypothesis  (main   (qualifier   !   has   been
associated with!)
                              learning-ability)
                        (secondary none)
                        (additional ! none!))
            (hypothesis  (main  (qualifier ! possible  trends
for!) paranoid)
                        (secondary none)
                        (additional
                         !  arithmetic ) verbal mean + 2  and
```

```
pict. completion ) performance mean + 2!))
              (hypothesis (main (qualifier ! possible!)
                                antisocial-personality-
disorder)
                          (secondary none)
                          (additional ! none!))
              (hypothesis   (main   (qualifier   !   possible!)
psychosis)
                          (secondary  (qualifier  !  conditions
for!)
                                     schizophrenia-or-
preschizophrenia)
                          (additional
                           !  picture completion (  performance
mean - 2!))
              (hypothesis  (main (qualifier ! tendencies  for!)
paranoid)
                          (secondary none)
                          (additional ! none!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary    (qualifier    !    acute
paranoid!)
                                     schizophrenia)
                          (additional ! none!)))
       (rule 20
              (score (lessp v (difference vmean 2))
                     (! vocabulary ( verbal mean - 2!))
              (hypothesis  (main  (qualifier !  lasting  effect
of!)
                                impoverished-early-environment)
                          (secondary none)
                          (additional ! none!))
              (hypothesis (main (qualifier ! none!) neurosis)
                          (secondary  (qualifier  !  conditions
for!)
                                     neurasthenia-or-hysteria)
                          (additional ! none!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary  (qualifier !  other  than
paranoid!)
                                     schizophrenia)
                          (additional  !  digit span  )  verbal
mean + 2!))
              (hypothesis (main (qualifier ! none!)
                                endogenous-mental-deficiency)
                          (secondary none)
                          (additional ! none!))
              (hypothesis   (main   (qualifier   !   possible!)
organic-brain-syndrome)
                          (secondary (qualifier ! sometimes!)
                                     local-lesions-in-dominant-
hemisphere-or-subordinate-temporal-lobe)
                          (additional ! none!))
              (hypothesis (main (qualifier ! none!)
                                cyclothymic-or-inadequate-
```

```
personality)
                              (secondary none)
                              (additional ¦ none¦)))
        (rule 21
              (score (greaterp v (add vmean 2))
                     (¦ vocabulary ) verbal mean + 2¦))
              (hypothesis (main (qualifier ¦ tendencies for¦)
                                obsessive-compulsive-or-
intellectualizing)
                              (secondary none)
                              (additional ¦ none¦))
              (hypothesis   (main   (qualifier   ¦   possible¦)
psychosis)
                              (secondary  (qualifier  ¦  conditions
for¦)
                                         schizophrenia-or-
ideational-preschizophrenia)
                              (additional ¦ none¦)))
        (rule 22
              (score (lessp pa (difference pmean 2))
                     (¦  picture arrangement ( performance mean
- 2¦))
              (hypothesis (main (qualifier ¦ likelihood of¦)
                                impulsivity-in-interpersonal-
relations-or-poor-planning)
                              (secondary (qualifier ¦ or likelihood
of¦)
                                         impaired-ability-in-
getting-along-with-others)
                              (additional ¦ none¦))
              (hypothesis (main (qualifier ¦ none¦)
                                normals-from-poor-cultural-
backgrounds)
                              (secondary   (qualifier   ¦    none¦)
lacking-sophistication)
                              (additional ¦ none¦))
              (hypothesis  (main (qualifier ¦ conditions  for¦)
depression)
                              (secondary (qualifier ¦ none¦)
                                         preoccupation-with-death)
                              (additional ¦ none¦))
              (hypothesis   (main   (qualifier   ¦   possible¦)
organic-brain-syndrome)
                              (secondary        (qualifier        ¦
particularly¦)
                                         right-hemisphere-or-
difuse-disfunctioning)
                              (additional    ¦   block   design   (
performance mean - 2¦))
              (hypothesis   (main   (qualifier   ¦   possible¦)
organic-brain-syndrome)
                              (secondary (qualifier ¦ perhaps¦)
                                         right-frontal-lobes-
disfunctioning)
                              (additional   ¦   block   design   (
```

```
performance mean - 2!))
              (hypothesis  (main (qualifier ! tendencies  for!)
procrastination)
                           (secondary none)
                           (additional ! none!))
              (hypothesis  (main (qualifier ! probable  setting
for!) psychosis)
                           (secondary  (qualifier  !  conditions
for!)
                                      schizophrenia)
                           (additional ! none!))
              (hypothesis    (main   (qualifier   !    possible
tendencies for!)
                                 obsessive-compulsive)
                           (secondary none)
                           (additional ! none!))
              (hypothesis    (main   (qualifier   !    possible
conditions for!)
                                 psychosomatic-migraine-or-
epileptic)
                           (secondary none)
                           (additional ! none!))
              (hypothesis (main (qualifier ! none!)
                                 intellectualizing-neurotic-or-
paranoid)
                           (secondary none)
                           (additional ! none!)))
        (rule 23
              (score (greaterp pa (add pmean 2))
                     (!  picture arrangement ) performance mean
+ 2!))
              (hypothesis  (main (qualifier ! none!)  socially-
adept-and-punctual)
                           (secondary  (qualifier  !   perhaps!)
normally-adjusting)
                           (additional ! none!))
              (hypothesis (main (qualifier ! none!)
                                 favorable-prognosis-with-
therapy)
                           (secondary none)
                           (additional ! none!))
              (hypothesis (main (qualifier ! none!)
                                 antisocial-personality-
disorder)
                           (secondary (qualifier ! none!)
                                      delinquency-or-scheming-
without-regard-to-consequences)
                           (additional ! performance IQ ) verbal
IQ + 9!))
              (hypothesis    (main   (qualifier   !    possible
tnedencies for!)
                                 paranoid)
                           (secondary none)
                           (additional ! none!)))
        (rule 24
```

```
                (score (lessp pc (difference pmean 2))
                       (! picture completion < performance mean -
2!))
                (hypothesis (main (qualifier
                                    !    emotional    disturbances
interfere with!)
                                  ability-to-distinguish-
essential-from-nonessential-details)
                            (secondary none)
                            (additional ! none!))
                (hypothesis (main (qualifier ! poor basic!)
                                  perceptual-and-conceptual-
ability-and-visual-concentration)
                            (secondary none)
                            (additional ! none!))
                (hypothesis (main (qualifier ! none!) anxiety)
                            (secondary none)
                            (additional ! none!))
                (hypothesis   (main   (qualifier   !   possible!)
psychosis)
                            (secondary  (qualifier  !  conditions
for!)
                                       schizophrenia)
                            (additional ! none!))
                (hypothesis   (main   (qualifier   !   possible!)
organic-brain-syndrome)
                            (secondary  (qualifier !  including!)
alcoholism)
                            (additional ! none!))
                (hypothesis (main (qualifier ! none!) psychosis)
                            (secondary    (qualifier    !    none!)
depression)
                            (additional ! none!))
                (hypothesis (main (qualifier ! none!) psychosis)
                            (secondary   (qualifier   !    none!)
schizophrenia)
                            (additional ! none!)))
        (rule 25
                (score (greaterp pc (add pmean 2))
                       (! picture completion > performance mean +
2!))
                (hypothesis   (main  (qualifier  !  greater  than
average !)
                                  breadth-of-general-information-
and-satisfactory-adjusting)
                            (secondary none)
                            (additional ! none!))
                (hypothesis (main (qualifier ! adequate!)
                                  ability-to-differentiate-
essential-from-nonessential-details)
                            (secondary none)
                            (additional ! none!))
                (hypothesis (main (qualifier ! possible!)
                                  antisocial-personality-
disorder)
```

```
                        (secondary  (qualifier  !  tendencies
for!) delinquent)
                        (additional ! none!))
            (hypothesis   (main  (qualifier  !  setting   for
endogenous type!)
                              mental-retardation)
                        (secondary none)
                        (additional ! none!))
            (hypothesis (main (qualifier ! none!)
                              favorable-prognosis-in-therapy)
                        (secondary none)
                        (additional ! none!))
            (hypothesis (main (qualifier ! adequate!)
                              perception-with-poor-visual-
motor-ability)
                        (secondary none)
                        (additional   !   block   design    (
performance mean - 2!))
            (hypothesis (main (qualifier ! tendencies for!)
                              obsessive-compulsive)
                        (secondary none)
                        (additional ! none!))
            (hypothesis  (main (qualifier ! none!)  paranoid-
over-alertness)
                        (secondary none)
                        (additional ! none!)))
      (rule 26
            (score (lessp oa (difference pmean 2))
                   (!  object  assembly ( performance mean  -
2!))
            (hypothesis  (main (qualifier !  likelihood  of!)
anxiety-or-tension)
                        (secondary   (qualifier   !   possible
associated with!)
                                    bodily-concern-or-
castration-anxiety)
                        (additional ! none!))
            (hypothesis  (main  (qualifier  !  likelihood  of
tendencies for!)
                              depression)
                        (secondary   (qualifier   !   perhaps
with!) aboulia)
                        (additional ! none!))
            (hypothesis   (main   (qualifier   !   possible!)
organic-brain-syndrome)
                        (secondary  (qualifier ! particularly
to!)
                                    right-hemisphere)
                        (additional ! none!))
            (hypothesis   (main   (qualifier   !   possible!)
psychosis)
                        (secondary (qualifier
                                    !  especially  conditions
for accute or anxious type!)
                                    schizophrenia)
```

```
                        (additional ! none!))
            (hypothesis (main (qualifier ! tendencies to be!)
hyperactive)
                        (secondary none)
                        (additional ! none!))
            (hypothesis   (main   (qualifier   !   possible!)
neurasthenia)
                        (secondary none)
                        (additional ! none!)))
      (rule 27
            (score (greaterp oa (add pmean 2))
                   (! object assembly ) performance mean +
2!))
            (hypothesis (main (qualifier ! good!)
                              perceptual-motor-coordination)
                        (secondary none)
                        (additional ! none!))
            (hypothesis  (main  (qualifier ! possibly  good!)
creative-ability)
                        (secondary none)
                        (additional ! none!))
            (hypothesis   (main   (qualifier   !   favorable!)
prognosis-for-therapy)
                        (secondary none)
                        (additional ! none!))
            (hypothesis   (main   (qualifier   !   possible!)
delinquency)
                        (secondary (qualifier ! possible!)
                                   disciplinary-problems)
                        (additional ! none!))
            (hypothesis   (main   (qualifier   !   setting   for
endogenous type!)
                              mental-retardation)
                        (secondary none)
                        (additional ! none!))
            (hypothesis   (main   (qualifier   !   possible!)
psychosis)
                        (secondary   (qualifier   !   possible
conditions for!)
                                   schizophrenia-or-
preschizophrenia)
                        (additional ! none!)))
      (rule 28
            (score (lessp bd (difference pmean 2))
                   (! block design ( performance mean - 2!))
            (hypothesis  (main  (qualifier ! none!)  organic-
brain-syndrome)
                        (secondary (qualifier ! none!)
                                   right-hemisphere-or-rignt-
parietal-lobe)
                        (additional ! none!))
            (hypothesis  (main (qualifier !  likelihood  of!)
anxiety)
                        (secondary none)
                        (additional ! none!))
```

```
                (hypothesis  (main  (qualifier ! likelihood  of!)
stress-or-tension)
                             (secondary none)
                             (additional ! none!))
                (hypothesis (main (qualifier ! tendencies for!)
                                  hyperactivity-or-impulsive)
                             (secondary none)
                             (additional ! none!))
                (hypothesis (main (qualifier ! none!) depression)
                             (secondary    (qualifier    !    and!)
aboulia)
                             (additional ! none!))
                (hypothesis    (main   (qualifier   !    possible
excessive!)
                                  repression-insecurity-or-
compulsivity)
                             (secondary none)
                             (additional ! none!))
                (hypothesis (main (qualifier ! possible!)
                                  antisocial-personality-
disorder)
                             (secondary (qualifier ! tendencies to
be!) delinquent)
                             (additional ! none!))
                (hypothesis    (main   (qualifier   !    possible
conditions to be!)
                                  paraniod)
                             (secondary none)
                             (additional ! none!)))
        (rule 29
                (score (greaterp bd (add pmean 2))
                       (! block design ) performance mean + 2!))
                (hypothesis (main (qualifier ! superior !)
                                  visual-motor-coordination-and-
perceptual-organization)
                             (secondary none)
                             (additional ! none!))
                (hypothesis (main (qualifier ! none!)  favorable-
therapy-prognosis)
                             (secondary none)
                             (additional ! none!))
                (hypothesis  (main  (qualifier ! possibly  good!)
creative-ability)
                             (secondary none)
                             (additional ! none!))
                (hypothesis  (main (qualifier !  contraindication
of !)
                                  organic-brain-syndrome)
                             (secondary none)
                             (additional ! none!))
                (hypothesis  (main (qualifier ! possible  setting
of!) psychosis)
                             (secondary  (qualifier  !  conditions
for!)
                                        schizophrenia-or-
```

```
preschizophrenia)
                        (additional ! none!))
             (hypothesis  (main  (qualifier  !  setting  for
endogenous type!)
                              mental-retardation)
                        (secondary none)
                        (additional ! none!)))
       (rule 30
             (score (lessp cd (difference pmean 2))
                    (! digit symbol ( performance mean - 2!))
             (hypothesis (main (qualifier ! probability of !)
                              organic-brain-syndrome)
                        (secondary       (qualifier       !
particularly!)
                                   right-hemisphere-or-motor-
area)
                        (additional ! none!))
             (hypothesis  (main  (qualifier  !  tendencies
toward!) depression)
                        (secondary    (qualifier    !    and
tendencies toward!)
                                   aboulia)
                        (additional ! none!))
             (hypothesis (main (qualifier ! probability of!)
                              anxiety-frustration-and-
tension)
                        (secondary none)
                        (additional ! none!))
             (hypothesis (main (qualifier ! none!) neurosis)
                        (secondary (qualifier
                                    !  possibly  anxiety  or
hysteria with!)
                                   neurotic-or-inadequate-
personality)
                        (additional ! none!))
             (hypothesis  (main  (qualifier  !  possible!)
psychosis)
                        (secondary (qualifier ! none!)
                                   dissociative-or-schizoid-
processes)
                        (additional ! none!))
             (hypothesis (main (qualifier ! possible!)
                              hyperactive-or-manic-
tendencies)
                        (secondary none)
                        (additional ! none!))
             (hypothesis   (main   (qualifier   !   possible
conditions to be!)
                              beat-downtrodden-homeless)
                        (secondary none)
                        (additional ! none!))
             (hypothesis (main (qualifier ! possible condition
to be!)
                              epileptic)
                        (secondary none)
```

```
                    (additional ! none!))
          (hypothesis (main (qualifier ! none!) obsessive-
doubting)
                    (secondary none)
                    (additional ! none!))
          (hypothesis (main (qualifier ! none!) psychosis)
                    (secondary (qualifier ! regressed!)
schizophrenia)
                    (additional ! none!))
          (hypothesis (main (qualifier ! possible!)
psychosis)
                    (secondary (qualifier ! none!)
depression)
                    (additional ! none!)))
     (rule 31
          (score (greaterp cd (add pmean 2))
                 (! digit symbol ) performance mean + 2!))
          (hypothesis (main (qualifier ! relatively!)
                         high-psychomotor-speed-and-
visual-motor-dexterity)
                    (secondary none)
                    (additional ! none!))
          (hypothesis (main (qualifier ! possibility of
relatively high!)
                         rote-learning-ability-or-
visual-memory)
                    (secondary none)
                    (additional ! none!))
          (hypothesis (main (qualifier ! relative!)
                         freedom-from-distractibility)
                    (secondary none)
                    (additional ! none!))
          (hypothesis (main (qualifier ! contraindication
of!)
                         low-energy-level)
                    (secondary none)
                    (additional ! none!))
          (hypothesis (main (qualifier ! possible!)
psychosis)
                    (secondary (qualifier ! none!)
                              schizophrenic-blandness)
                    (additional ! none!)))
     (rule 32
          (score (greaterp piq 110) (! performance IQ )
110!))
          (certain psychosis)
          (hypothesis (main (qualifier ! possibly!)
                         antisocial-personality-
disorder)
                    (secondary none)
                    (additional ! none!)))
     (rule 33
          (score (greaterp cm (add vmean 2))
                 (! comprehension ) verbal mean - 2!))
          (hypothesis (main (qualifier ! none!) strict-
```

```
moral-code)
                              (secondary none)
                              (additional ! none!)))
        (rule 34
              (score (lessp cm (difference vmean 2))
                     (! comprehension ( verbal mean - 2!))
              (hypothesis (main (qualifier ! may reflect!)
                                inflexibility-and-impaired-
judgement)
                              (secondary none)
                              (additional ! none!)))
        (rule 35
              (score (lessp ds (difference v 2))
                     (! digit span ( vocabulary - 2!))
              (hypothesis (main (qualifier ! none!) anxiety)
                              (secondary none)
                              (additional ! none!)))
        (rule 36
              (score (lessp ds (difference v 7))
                     (! digit span ( vocabulary - 7!))
              (hypothesis (main (qualifier ! possible!)
                                incipient-psychotic-break)
                              (secondary none)
                              (additional ! none!)))
        (rule 37
              (score (and (or (greaterp ds v) (eq ds v))
                          (or (greaterp ds a) (eq ds a)))
                     (! digit span )= arithmetic or digit  span
)= vocabulary!))
              (hypothesis   (main   (qualifier   !   possible!)
psychosis)
                              (secondary   (qualifier   !   possible
tendency for!)
                                          schizoid-or-schizophrenia)
                              (additional ! none!)))
        (rule 38
              (score (lessp ds (difference i 5))
                     (! digit span ( information - 5!))
              (hypothesis (main (qualifier ! none!) anxiety)
                              (secondary none)
                              (additional ! none!)))
        (rule 39
              (score  (greaterp  ds (add a 2)) (! digit span  )
arithemetic + 2!))
              (hypothesis (main (qualifier ! none!)
                                antisocial-personality-
disorder)
                              (secondary none)
                              (additional ! none!))
              (hypothesis   (main   (qualifier   !   none!)   low-
anxiety-tolerance)
                              (secondary none)
                              (additional ! none!)))
        (rule 40
              (score (lessp ds (difference vmean 6))
```

```
                    (! digit span < verbal mean - 6!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary   (qualifier   !   anxious
undifferentiated!)
                                     schizophrenia)
                          (additional ! none!)))
        (rule 41
              (score (lessp i (difference v 2))
                     (! information < vocabulary - 2!))
              (hypothesis (main (qualifier ! none!) depression)
                          (secondary none)
                          (additional ! none!))
              (hypothesis (main (qualifier ! none!) hysteria)
                          (secondary none)
                          (additional ! none!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary   (qualifier   !    none!)
schizophrenia)
                          (additional ! none!)))
        (rule 42
              (score (and (lessp i (difference cm 2))
                          (lessp i (difference s 2)))
                     (! information < comprehension - 2 and
information < vocabulary - 2!))
              (hypothesis  (main  (qualifier !  tendency  for!)
repression)
                          (secondary none)
                          (additional ! none!)))
        (rule 43
              (score (and (lessp ds (difference s 2))
                          (and (lessp v (difference s 2))
                               (and (lessp a (difference s 2))
                                    (lessp   i   (difference   s
2)))))
                     (! information, arithmetic, vocabulary,
and digit span < similarities + 2!))
              (hypothesis   (main   (qualifier   !   trend   for!)
paranoid)
                          (secondary none)
                          (additional ! none!)))
        (rule 44
              (score (lessp s (difference v 2))
                     (! similarities < vocabulary - 2!))
              (hypothesis (main (qualifier ! none!) depression)
                          (secondary none)
                          (additional ! none!))
              (hypothesis  (main  (qualifier ! none!)  organic-
brain-syndrome)
                          (secondary none)
                          (additional ! none!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary   (qualifier   !    none!)
schizophrenia)
                          (additional ! none!)))
        (rule 45
```

```
                (score (greaterp s 18) (! similarities ) 18!))
                (hypothesis    (main    (qualifier    !    none!)
intellectualization)
                                (secondary none)
                                (additional ! none!)))
        (rule 46
                (score  (and  (greaterp s (add v 2)) (greaterp  s
(add vmean 2)))
                        (!  similarities  ) verbal mean  +  2  and
vocabulary + 2!))
                (hypothesis  (main (qualifier ! tendency to  be!)
paranoid)
                                (secondary none)
                                (additional ! none!)))
        (rule 47
                (score (and (lessp piq 90) (greaterp viq 109))
                        (!  verbal  IQ ) 109 and performance IQ  (
90!))
                (hypothesis  (main  (qualifier !  possible  trend
toward!)
                                        depression)
                                (secondary none)
                                (additional ! none!)))
        (rule 48
                (score (greaterp viq (add piq 5))
                        (! verbal IQ ) performance IQ +5!))
                (hypothesis (main (qualifier ! none!) neurosis)
                                (secondary   (qualifier   !    none!)
depression)
                                (additional ! none!)))
        (rule 49
                (observe 2 1)
                (hypothesis (main (qualifier ! none!) depression)
                                (secondary (qualifier ! or!)
                                                neurotic-or-psychotic-
depression)
                                (additional ! none!))
                (hypothesis (main (qualifier ! none!) psychosis)
                                (secondary (qualifier
                                                !  paranoid or simple  or
undifferentiated!)
                                                schizophrenia)
                                (additional ! none!)))
        (rule 50
                (observe 2 2)
                (hypothesis (main (qualifier ! none!) psychosis)
                                (secondary    (qualifier    !    none!)
depression)
                                (additional ! none!))
                (hypothesis (main (qualifier ! none!) psychosis)
                                (secondary        (qualifier        !
undifferentiated!)
                                                schizophrenia)
                                (additional ! none!)))
        (rule 51
```

```
                (observe 2 3)
                (hypothesis (main (qualifier ! none!) depression)
                            (secondary none)
                            (additional ! none!))
                (hypothesis (main (qualifier ! none!) hysteria)
                            (secondary none)
                            (additional ! none!))
                (hypothesis (main (qualifier ! none!) psychosis)
                            (secondary   (qualifier   !    none!)
schizophrenia)
                            (additional ! none!)))
        (rule 52
                (observe 2 4)
                (hypothesis (main (qualifier ! none!) psychosis)
                            (secondary    (qualifier   !   cronic
undifferentiated!)
                                          schizophrenia)
                            (additional ! none!)))
        (rule 53
                (observe 2 5)
                (hypothesis  (main  (qualifier  !  none!)   self-
deprecating-attitude)
                            (secondary none)
                            (additional ! none!)))
        (rule 54
                (observe 2 6)
                (observe 5 4)
                (hypothesis (main (qualifier ! none!) psychosis)
                            (secondary (qualifier ! none!)
                                       schizophrenia-and-
depression)
                            (additional ! none!)))
        (rule 55
                (score (and ((= ds cm)
                            (and  ((= a cm) (and ((= s cm) ((=  i
cm))))
                        (! information,  arithmetic,  similarities
and digit span (= comprehension!))
                (hypothesis (main (qualifier ! none!) psychosis)
                            (secondary  (qualifier  !  paranoid!)
schizophrenia)
                            (additional ! none!)))
        (rule 56
                (score ()= ds prev-ds)
                       (!  current  digit span )= previous  digit
span!))
                (hypothesis (main (qualifier ! possibly may still
be!)
                                  impairment-of-recent-memory)
                            (secondary none)
                            (additional ! none!)))
        (rule 57
                (score (and (greaterp pc (add pmean 2))
                            (and (greaterp s (add vmean 2))
                                 (greaterp a (add vmean 2))))
```

```
                    (¦  airthmetic and similarities  ) verbal
mean + 2¦
                     ¦              and  picture  completion  )
performance mean + 2¦))
               (hypothesis (main (qualifier ¦ none¦) paranoid)
                           (secondary none)
                           (additional ¦ none¦)))
         (rule 58
               (observe 2 9)
               (observe 5 5)
               (observe 5 6)
               (hypothesis (main (qualifier ¦ none¦)
                                 antisocial-personality-
disorder)
                           (secondary none)
                           (additional ¦ none¦)))
         (rule 59
               (observe 2 10)
               (score  (greaterp s (add v 2)) (¦ similarities  )
vocabulary + 2¦))
               (hypothesis (main (qualifier ¦ none¦) neurosis)
                           (secondary    (qualifier   ¦   none¦)
depression)
                           (additional ¦ none¦)))
         (rule 60
               (score (and (greaterp piq 100) (greaterp cm  (add
i 2)))
                      (¦  comprehension  ) information +  2  and
performance IQ ) 100¦))
               (hypothesis  (main  (qualifier  ¦  features  of¦)
hysteria)
                           (secondary none)
                           (additional ¦ none¦)))
         (rule 61
               (score (and ()= ds vmean)
                           (and (lessp cm (difference vmean 2))
                                (lessp a (difference vmean 2))))
                      (¦  arithmetic and comprehension (  verbal
mean - 2¦
                       ¦              and  digit span  )=  verbal
mean¦))
               (hypothesis (main (qualifier ¦ none¦) psychosis)
                           (secondary   (qualifier   ¦   acute¦)
schizophrenic-episode)
                           (additional ¦ none¦))
               (hypothesis (main (qualifier ¦ none¦) psychosis)
                           (secondary   (qualifier   ¦   chronic
undifferentiated¦)
                                        schizophrenia)
                           (additional ¦ none¦)))
         (rule 62
               (score (and (greaterp a (add ds 2))
                           (and (lessp ds (difference v 2))
                                (lessp a (difference v 2))))
                      (¦  arithmetic and digit span ( vocabulary
```

```
- 2 !
                        !            and arithmetic ) digit  span
+ 2!))
              (hypothesis (main (qualifier ! none!) neurosis)
                          (secondary none)
                          (additional ! none!)))
        (rule 63
              (score  (and  (lessp pmean 85) (lessp bd (add  cm
2)))
                      (! block design ( performance mean - 2 !
                       !            and performance mean ( 85!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary   (qualifier   !    none!)
depression)
                          (additional ! none!)))
        (rule 64
              (score  (and (greaterp i (add cm 2)) (greaterp  v
(add cm 2)))
                      (!    vocabulary    and   information    )
comprehension + 2!))
              (hypothesis (main (qualifier ! none!) obsessive)
                          (secondary none)
                          (additional ! none!))
              (hypothesis (main (qualifier ! none!)  psychotic-
depressives)
                          (secondary none)
                          (additional ! none!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary none)
                          (additional ! none!)))
        (rule 65
              (score (and (lessp cm (difference vmean 2))
                          (lessp i (difference vmean 2)))
                      (!  information and comprehension ( verbal
mean - 2!))
              (hypothesis    (main    (qualifier    !    none!)
psychopathology)
                          (secondary none)
                          (additional ! none!)))
        (rule 66
              (score (and (lessp v (difference vmean 2))
                          (lessp i (difference vmean 2)))
                      (!  information  and vocabulary  (  verbal
mean - 2!))
              (hypothesis  (main  (qualifier !  tendency  for!)
repression)
                          (secondary none)
                          (additional ! none!)))
        (rule 67
              (score (and (greaterp pc (add pmean 2))
                          (and (lessp v (add vmean 2))
                               (greaterp viq (add piq 2))))
                      (!  verbal  IQ  ) performance IQ  +  2!  !
and vocabulary ( verbal mean + 2!
                                                              !
```

```
and picture completion ) performance mean + 2!))
                (hypothesis (main (qualifier ! none!) hysteria)
                            (secondary none)
                            (additional ! none!))
                (hypothesis (main (qualifier ! none!)  obsessive-
compulsive)
                            (secondary none)
                            (additional ! none!)))
          (rule 68
                (score (and ()= cd pa)
                            (and ()= oa pa)
                                 (and ()= pc pa) (greaterp v (add
pa 2)))))
                       (! vocabulary ) picture arrangement + 2! !
picture completion, object assembly, and digit symbol!
                                                                  !
are each )= picture arrangement!))
                (hypothesis (main (qualifier ! none!) depression)
                            (secondary none)
                            (additional ! none!)))
          (rule 69
                (score   (and  (and  ()=  pa  pmean)  (lessp   pc
(difference pmean 2)))
                            (lessp a (difference vmean 2)))
                       (!   arithmetic  (  verbal  mean  - 2!   !
and picture completion ( performance mean - 2!
                                                          !
and picture arrangement )= performance mean!))
                (hypothesis (main (qualifier ! none!)
                                  antisocial-personality-
disorder)
                            (secondary  (qualifier !  including!)
addiction)
                            (additional ! none!)))
          (rule 70
                (observe 2 10)
                (score (and (lessp s (difference vmean 2))
                            (lessp s (difference v 1)))
                       (! similarities ( vocabulary - 1!
                        !              and  similarities (  verbal
mean - 2!))
                (hypothesis (main (qualifier ! none!) psychosis)
                            (secondary   (qualifier   !    none!)
depression)
                            (additional ! none!)))
          (rule 71
                (score (and (greaterp v (add vmean 2))
                            (and (greaterp cm (add vmean 2))
                                 (greaterp i (add vmean 2))))
                       (!   information,    comprehension,    and
vocabulary!
                        !                 are each ) verbal mean +
2!))
                (hypothesis    (main    (qualifier    !    none!)
intellectual-striving)
```

```
                        (secondary none)
                        (additional ¦ none¦)))
        (rule 72
              (observe 2 11)
              (observe 3 1)
              (observe 2 12)
              (score  (greaterp  fsiq 90) (¦ full  scale  IQ  )
90¦))
              (hypothesis (main (qualifier ¦ none¦) psychosis)
                        (secondary (qualifier
                                    ¦  fantasies rather  than
delusions and latent¦)
                                   schizophrenia)
                        (additional ¦ none¦)))
        (rule 73
              (score (and (greaterp pc (add pmean 2))
                        (greaterp a (add vmean 2)))
                     (¦ arithmetic ) verbal mean + 2¦
                      ¦              and  picture  completion  )
performance mean + 2¦))
              (hypothesis   (main   (qualifier   ¦   possible¦)
paranoid-alertness)
                        (secondary none)
                        (additional ¦ none¦)))
        (rule 74
              (score (and (greaterp pc (add pmean 2))
                        (and (greaterp s (add vmean 2))
                             (greaterp a (add vmean 2))))
                     (¦  airthmetic  and similarities )  verbal
mean + 2¦
                      ¦              and  picture  completion  )
performance mean + 2¦))
              (hypothesis (main (qualifier ¦ none¦) projective-
trend)
                        (secondary none)
                        (additional ¦ none¦)))
        (rule 75
              (score (and (greaterp prev-piq piq) (greaterp  bd
(add pmean 2)))
                     (¦ block design ) performance mean + 2¦
                      ¦              and previous performance IQ
) current performance IQ¦))
              (hypothesis (main (qualifier ¦ none¦) psychosis)
                        (secondary  (qualifier  ¦  paranoid¦)
schizophrenia)
                        (additional ¦ none¦)))
        (rule 76
              (score  (and  (lessp piq 90)  (greaterp  bd  (add
pmean 2)))
                     (¦ block design ) performance mean + 2¦
                      ¦             and performance IQ ( 90¦))
              (hypothesis (main (qualifier ¦ none¦) psychosis)
                        (secondary    (qualifier    ¦    none¦)
schizophrenia)
                        (additional ¦ none¦)))
```

```
        (rule 77
              (score (and (greaterp piq 110) (greaterp ds  (add
vmean 2)))
                     (¦ digit span ) verbal mean + 2¦
                      ¦            and performance IQ ) 110¦))
              (hypothesis  (main (qualifier ¦ none¦) character-
disorders)
                           (secondary none)
                           (additional ¦ none¦))
              (hypothesis (main (qualifier ¦ none¦) hysteria)
                           (secondary none)
                           (additional ¦ none¦)))
        (rule 78
              (score (and (greaterp ds (add vmean 2))
                           (lessp a (difference vmean 2)))
                     (¦ arithmetic ( verbal mean - 2¦
                      ¦            and digit span ) verbal mean
- 2¦))
              (hypothesis (main (qualifier ¦ none¦)
                                schizoid-personality-or-
character)
                           (secondary none)
                           (additional ¦ none¦)))
        (rule 79
              (score (greaterp ds (add vmean 2))
                     (¦ digit span ) verbal mean + 2¦))
              (certain neurosis)
              (hypothesis (main (qualifier ¦ none¦) ideational-
tendendcies)
                           (secondary none)
                           (additional ¦ none¦)))
        (rule 80
              (score (or (and (greaterp viq (add piq 10))
                              (and (lessp cm (difference  vmean
2))
                                   (and  (greaterp v (add vmean
2))
                                         (greaterp i (add  vmean
2)))))
                         (lessp s vmean))
                     (¦   similarities   (   verbal   mean¦   ¦
or information and vocabulary ) verbal mean + 2¦
                                                          ¦
and comprehension ( verbal mean - 2¦
                                                          ¦
and verbal IQ ) performance IQ¦))
              (hypothesis (main (qualifier ¦ none¦) neurosis)
                           (secondary    (qualifier   ¦   none¦)
obsessive-compulsive)
                           (additional ¦ none¦)))
        (rule 81
              (observe 3 2)
              (score (greaterp pa (add pmean 2))
                     (¦ picture arrangement ) performance  mean
+ 2¦))
```

```
                (hypothesis (main (qualifier ! tendency for!)
                                  antisocial-personality-
disorder)
                            (secondary none)
                            (additional ! none!)))
          (rule 82
                (observe 2 13)
                (observe 3 3)
                (observe 3 9)
                (score (and (greaterp pc (add pmean 2))
                            (and (greaterp a (add vmean 2))
                                 (greaterp s (add vmean 2))))
                       (! similarities and arithmetic ) verbal
mean + 2!
                        !              and picture completion )
performance mean + 2!))
                (hypothesis (main (qualifier ! none!) psranoid-
personality)
                            (secondary none)
                            (additional ! none!)))
          (rule 83
                (score (and (greaterp pc (add pmean 2))
                            (greaterp s (add vmean 2)))
                       (! similarities ) verbal mean + 2!
                        !              and picture completion )
performance mean + 2!))
                (hypothesis (main (qualifier ! none!) psychosis)
                            (secondary (qualifier ! paranoid!)
schizophrenia)
                            (additional ! none!)))
          (rule 84
                (observe 3 10)
                (observe 3 11)
                (observe 5 5)
                (score (and (greaterp bd (add pmean 2))
                            (greaterp ds (add vmean 2)))
                       (! block design ) verbal mean + 2 and
performance mean + 2!))
                (hypothesis (main (qualifier ! none!)
                                  antisocial-personality-
disorder)
                            (secondary none)
                            (additional ! none!)))
          (rule 85
                (score (and (greaterp bd (add pmean 2))
                            (greaterp bd (add vmean 2)))
                       (! block design ) verbal mean + 2 and
performence mean + 2!))
                (hypothesis (main (qualifier ! none!) schizoid)
                            (secondary none)
                            (additional ! none!))
                (hypothesis (main (qualifier ! none!) psychosis)
                            (secondary (qualifier ! none!)
schizophrenia)
                            (additional ! none!)))
```

```
        (rule 86
              (score (and (greaterp ds (add vmean 2))
                          (greaterp a (add vmean 2)))
                     (! arithmetic and digit span ) verbal mean
+ 2!))
              (hypothesis   (main   (qualifier   !   possible!)
obsessive-compulsive)
                          (secondary  (qualifier ! or  tendency
for!) schizoid)
                          (additional ! none!)))
        (rule 87
              (score (and (lessp v (difference vmean 2))
                          (greaterp i (add vmean 2)))
                     (! information ) verbal mean + 2 !
                      !              and vocabulary ( verbal mean
- 3!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary  (qualifier  !  simple  or
latent!)
                                      schizophrenia)
                          (additional ! none!)))
        (rule 88
              (score (and (lessp ds (difference vmean 2))
                          (and (lessp s (difference vmean 2))
                               (and (lessp cm (difference vmean
2))
                                    (and (greaterp v (add vmean
2))
                                         (greaterp i (add vmean
2))))))
                     (!  information  and vocabulary  )  verbal
mean  + 2! !           and  comprehension,  similarities,  and
digit symbol !
                                                              !
are each ( verbal mean - 2!))
              (hypothesis  (main (qualifier ! none!) obsessive-
compulsive)
                          (secondary none)
                          (additional ! none!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary   (qualifier   !    none!)
depression)
                          (additional ! none!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary        (qualifier        !
undifferentiated!)
                                      schizophrenia)
                          (additional ! none!)))
        (rule 89
              (score (and (lessp cd (difference pmean 2))
                          (and (lessp bd (difference pmean 2))
                               (and (lessp pa (difference pmean
2))
                                    (greaterp   oa   (add   pmean
2)))))
```

```
                    (¦ object assembly ) performance mean + 2¦
¦            and picture assembly, block design, digit symbol ¦
                                                               ¦
are each ( performance mean - 2¦))
              (hypothesis (main (qualifier ¦ none¦) blandness)
                          (secondary none)
                          (additional ¦ none¦))
              (hypothesis (main (qualifier ¦ none¦) psychosis)
                          (secondary   (qualifier   ¦    none¦)
schizophrenia)
                          (additional ¦ none¦)))
        (rule 90
              (score (and (lessp pc (difference pmean 2))
                          (and (greaterp viq (add piq 5))
                               (greaterp v (add vmean 2))))
                     (¦   vocabulary  )  verbal  mean  +  2¦ ¦
and verbal IQ ) performance IQ +5¦
                                                          ¦
and picture completion ( performance mean - 2¦))
              (hypothesis (main (qualifier ¦ none¦) psychosis)
                          (secondary   (qualifier   ¦    none¦)
intellectualizing)
                          (additional ¦ none¦)))
        (rule 91
              (observe 3 5)
              (observe 3 12)
              (score (and (greaterp oa (add pmean 2))
                          (greaterp bd (add pmean 2)))
                     (¦  block  design  and object  assembly  )
performance mean + 2¦))
              (hypothesis (main (qualifier ¦ none¦) psychosis)
                          (secondary    (qualifier   ¦    acute
paranoid¦)
                                        schizophrenia)
                          (additional ¦ none¦)))
        (rule 92
              (observe 3 6)
              (observe 4 13)
              (observe 4 12)
              (score  (greaterp  fsiq 110) (¦ full scale  IQ  )
110¦))
              (hypothesis (main (qualifier ¦ none¦) psychosis)
                          (secondary   (qualifier   ¦   simple¦)
schizophrenia)
                          (additional ¦ none¦)))
        (rule 93
              (observe 3 7)
              (observe 4 11)
              (observe 4 12)
              (observe 5 1)
              (observe 5 2)
              (observe 5 3)
              (observe 5 7)
              (hypothesis (main (qualifier ¦ none¦) depression)
                          (secondary none)
```

```
                        (additional ! none!)))
        (rule 94
              (observe 5 2)
              (observe 5 3)
              (hypothesis (main (qualifier ! none!) paranoia)
                          (secondary none)
                          (additional ! none!)))
        (rule 95
              (score (and (lessp ds 2)
                          (and (lessp cm (difference vmean 2))
                               (and (lessp a (difference  vmean
2))
                                    (lessp  s (difference vmean
2)))))
                     (!         similarities,         arithmetic,
comprehension, and digit span!
                      !                are each ( verbal mean  -
2!))
              (hypothesis (main (qualifier ! none!) depression)
                          (secondary none)
                          (additional ! none!)))
        (rule 96
              (observe 4 1)
              (observe 5 8)
              (observe 5 9)
              (hypothesis (main (qualifier ! none!)
                                antisocial-personality-
disorder)
                          (secondary none)
                          (additional ! none!)))
        (rule 97
              (score (and (greaterp cm (add vmean 2))
                          (lessp a (difference vmean 2)))
                     (! arithmetic ( verbal mean - 2!
                      !              and comprehension )  verbal
mean + 2!))
              (hypothesis  (main (qualifier ! contraindicates!)
psychosis)
                          (secondary        (qualifier        !
conrtraindicates!)
                                            depression)
                          (additional ! none!)))
        (rule 98
              (score (and (greaterp piq viq)
                          (and (greaterp piq 100)
                               (lessp i (difference cm 2))))
                     (!  information  (  comprehension  - 2!  !
and performance IQ ) 100!
                                                              !
and performance IQ ) verbal IQ!))
              (hypothesis  (main  (qualifier  !  features  of!)
hysteria)
                          (secondary none)
                          (additional ! none!)))
        (rule 99
```

```
                (observe 4 2)
                (hypothesis (main (qualifier ! none!) psychosis)
                            (secondary   (qualifier   !    none!)
schizophrenia)
                            (additional ! none!)))
        (rule 100
                (observe 4 3)
                (hypothesis   (main   (qualifier   !   possible!)
psychosis)
                            (secondary    (qualifier   !   cronic
undifferentiated!)
                                        schizophrenia)
                            (additional ! none!)))
        (rule 101
                (score (and (lessp oa (difference pmean 2))
                            (lessp ds (difference vmean 2)))
                       (! digit span < verbal mean - 2!
                        !              and  object   assembly   <
performance mean - 2!))
                (hypothesis (main (qualifier ! none!) depression)
                            (secondary none)
                            (additional ! none!))
                (hypothesis (main (qualifier ! none!) tension)
                            (secondary none)
                            (additional ! none!)))
        (rule 102
                (score (and (lessp pc (difference pmean 2))
                            (lessp oa (difference pmean 2)))
                       (!  object assembly and picture assembly <
performance mean - 2!))
                (hypothesis (main (qualifier ! none!) psychosis)
                            (secondary (qualifier
                                        !  episode of  or  cronic
undifferentiated!)
                                        schizophrenia)
                            (additional ! none!)))
        (rule 103
                (score (and (lessp s (difference vmean 2)) (lessp
piq 90))
                       (! performance IQ < 90 !
                        !             and  similarities <  verbal
mean - 2!))
                (hypothesis (main (qualifier ! none!) psychosis)
                            (secondary  (qualifier  !  paranoid!)
schizophrenia)
                            (additional ! none!)))
        (rule 104
                (score (and (lessp s (difference vmean 2))
                            (lessp pa (difference pmean 2)))
                       (! picture arrangement < performance  mean
- 2!
                        !             and  similarities <  verbal
mean - 2!))
                (hypothesis (main (qualifier ! none!) psychosis)
                            (secondary   (qualifier   !    cronic
```

```
undifferentiated!)
                                        schizophrenia)
                              (additional ! none!)))
        (rule 105
              (score (and (greaterp cd 10)
                          (and (greaterp oa 10)
                               (lessp v (difference vmean 2))))
                     (! vocabulary ( verbal mean - 2! ! and object assembly ) 10!
                                                                    !
and digit symbol ) 10!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary (qualifier ! simple!) schizophrenia)
                          (additional ! none!)))
        (rule 106
              (score (and (greaterp ps 5) (greaterp vs 5))
                     (! verbal and performance scatter ) 5!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary (qualifier ! tendency for!) schizophrenia)
                          (additional ! none!)))
        (rule 107
              (observe 4 4)
              (score (and (greaterp oa (difference pmean 4))
                          (and (lessp oa (difference pmean 1))
                               (and (greaterp bd (difference pmean 4))
                                    (and (lessp bd (difference pmean 1))
                                         (and (greaterp cd 7)
                                              (and (greaterp pa 7)
                                                   (greaterp pc 7)))))))
                     (! picture completion, picture arrangement, and digit symbol ) 7! ! and block design and object assembly ( performance mean - 1!
! and block design and object assembly ) performance mean - 4!))
              (hypothesis (main (qualifier ! none!) tension)
                          (secondary none)
                          (additional ! none!)))
        (rule 108
              (observe 4 5)
              (score (and (lessp pc (difference pmean 4))
                          (and (lessp a (difference vmean 4))
                               (lessp ds (difference vmean 4))))
                     (! digit span and arithmetic ( verbal mean - 4!
                      ! and picture completion ( performance mean - 4!))
              (hypothesis (main (qualifier ! none!) psychosis)
```

```
                        (secondary    (qualifier    !    none!)
schizophrenia)
                        (additional ! none!)))
        (rule 109
              (score (and ((= viq piq)
                        (and (greaterp pa (add pmean 2))
                             (lessp a (difference vmean 2))))
                    (!   arithmetic   (   verbal   mean   - 2!    !
and picture arrangement ) performance mean + 2!
                                                          !
and verbal IQ (= performance IQ!))
              (hypothesis (main (qualifier ! none!)
                                narcissistic-character-
disorder)
                        (secondary none)
                        (additional ! none!)))
        (rule 110
              (score (and ((= viq piq)
                        (and (greaterp cm (add vmean 2))
                             (lessp i (difference vmean 2))))
                    (!  information  (  verbal   mean   - 2!    !
and comprehension ) verbal mean + 2!
                                                           !
and verbal IQ (= performance IQ!))
              (hypothesis (main (qualifier ! none!) hysteria)
                        (secondary none)
                        (additional ! none!)))
        (rule 111
              (score (and (greaterp i (add pmean 2))
                        (greaterp a (add pmean 2)))
                    (!     arithmetic     and     information     )
performance mean + 2!))
              (hypothesis (main (qualifier ! none!) depression)
                        (secondary none)
                        (additional ! none!)))
        (rule 112
              (score (and (lessp piq 91)
                        (and (lessp viq 91) (greaterp v  (add
vmean 2))))
                    (! vocabulary ) verbal mean + 2!
                     !              and verbal and  performance
IQs are ) 91!))
              (hypothesis (main (qualifier ! none!) alcoholism)
                        (secondary none)
                        (additional ! none!)))
        (rule 113
              (observe 3 6)
              (score  (greaterp  fsiq 110) (! full scale  IQ  )
110!))
              (hypothesis (main (qualifier ! none!) psychosis)
                        (secondary    (qualifier    !    none!)
schizophrenia)
                        (additional ! none!)))
        (rule 114
              (score (and (lessp cm (difference vmean 2))
```

```
                              (and (lessp a (difference vmean 2))
                                   (and (greaterp cd (add pmean 2))
                                        (and   (greaterp  oa   (add
pmean 2))
                                             (and (greaterp bd (add
pmean 2))
                                                  (and (greaterp cd
                                                       (add
pa 2))

; ((((( start back on the left (((((
      (and (greaterp oa (add pa 2))
           (and (greaterp bd (add pa 2))
                (and (greaterp cd (add pc 2))
                     (and (greaterp oa (add pc 2)) (greaterp  bd
(add pc 2))))))
; ))))) continue on the right )))))
))))))
                         (! block design, object assembly, and
digit symbol ! !                  are each ) picture completion +
2!

!              and  block design,  object  assembly,  and  digit
symbol!

!                 are each ) picture arrangement + 2!

!              and  block design,  objece  assembly,  and  digit
symbol!

!                 are each ) performance mean + 2!

!              and arithmetic and comprehension ( verbal mean  -
2!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary    (qualifier   !   none!)
schizophrenia)
                          (additional ! none!)))
        (rule 115
              (score (and (lessp pa (difference pmean 2))
                          (and (lessp pc (difference pmean 2))
                               (and (lessp ds (difference vmean
2))
                                    (and   (greaterp  oa   (add
pmean 2))
B                                        (and  (greaterp   bd
(add pmean 2))
                                             (and (lessp ds
                                              (difference v 2))

; ((((( start back on the left (((((
      (and (lessp cm (difference v 2))
           (and (lessp a (difference v 2))
                (and   (lessp  s  (difference  v  2))  (lessp  i
(difference v 2)))))
```

```
; ))))) continue on the right )))))
))))))
                    (! information, similarities, arithmetic,
comprehension, and digit span! !                    are each (
vocabulary - 2!

!           and block design and object assembly ) performance
mean + 2!

!           and digit span ( verbal mean - 2!

!           and picture completion and picture arrangement!

!               are each ( performance mean - 2!))
            (hypothesis (main (qualifier ! none!) psychosis)
                        (secondary   (qualifier   !   acute!)
schizophrenic-episode)
                        (additional ! none!))
            (hypothesis (main (qualifier ! none!) psychosis)
                        (secondary   (qualifier   !   chronic
undifferentiated!)
                                    schizophrenia)
                        (additional ! none!)))
      (rule 116
            (score (and (lessp a (difference vmean 2))
                        (lessp s (difference vmean 2)))
                   (!  similarities and arithmetic  (  verbal
mean - 2!))
            (certain psychosis)
            (hypothesis    (main   (qualifier   !   chronic!)
schizophrenia)
                        (secondary none)
                        (additional ! none!)))
      (rule 117
            (score (greaterp cm (add vmean 2))
                   (! comprehension ) verbal mean + 2!))
            (certain psychosis)
            (hypothesis   (main   (qualifier   !   paranoid!)
schizophrenia)
                        (secondary none)
                        (additional ! none!)))
      (rule 118
            (score (and (greaterp i (add v 2))
                        (and (greaterp i (add s 2))
                             (and (greaterp cm (add v 2))
                                  (and (greaterp cm s)
                                       (and (greaterp cm 10)
                                            (greaterp       i
10))))))
                   (!  information and comprehension ) 10!  !
and comprehension ) similarities!
                                                          !
and comprehension ) vocabulary + 2!
                                                          !
and information ) similarities + 2!
```

```
                                                                    !
and information ) vocabulary  + 2!))
              (bckgrnd 5 10)
              (hypothesis    (main    (qualifier    !    none!)
intellectual-striving)
                          (secondary none)
                          (additional ! none!)))
        (rule 119
              (score (and (greaterp ds 10) (greaterp oa 10))
                     (! object assembly and digit span ) 10!))
              (certain psychosis)
              (hypothesis    (main    (qualifier    !    none!)
schizophrenia)
                          (secondary   (qualifier    !    none!)
blandness)
                          (additional ! none!)))
        (rule 120
              (score (and (lessp pc (difference pmean 2))
                          (lessp s (difference vmean 2)))
                     (! arithmetic ( verbal mean - 2!
                      !              and  picture  completion  (
performance mean - 2!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary    (qualifier    !    none!)
schizophrenia)
                          (additional ! none!)))
        (rule 121
              (score (and (lessp viq (difference piq 10))
                          (lessp v (difference vmean 2)))
                     (! vocabulary ( verbal mean - 2!
                      !             and verbal IQ (  performance
IQ - 10!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary   (qualifier    !    chronic
undifferentiated!)
                                       schizophrenia)
                          (additional ! none!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary    (qualifier    !    none!)
depression)
                          (additional ! none!)))
        (rule 122
              (score (and (lessp pa (difference pmean 2))
                          (and (lessp pc (difference pmean 2))
                               (lessp   ds   (difference   vmean
2))))
                     (! digit span ( verbal mean - 2!
                      !              and picture completion  and
arrangement are ( performance mean - 2!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary   (qualifier    !    none!)
schizophrenia)
                          (additional ! none!)))
        (rule 123
              (score (lessp i (difference cm 2))
```

```
                    (! information < comprehension - 2!))
              (certain psychosis)
              (hypothesis (main (qualifier ! none!) paranoid)
                          (secondary none)
                          (additional ! none!)))
        (rule 124
              (score (and (lessp vs 6)
                          (and  (greaterp  vs 3) (greaterp  viq
(add piq 5))))
                    (! verbal IQ > performance IQ + 5!
                     !              and  3 < verbal  scatter  <
6!))
              (hypothesis  (main  (qualifier  !  tendency  to!)
neurosis)
                          (secondary none)
                          (additional ! none!)))
        (rule 125
              (score (and (lessp pc (difference oa 3))
                          (and (lessp pc (difference bd 3))
                               (and  (lessp pa  (difference  oa
3))
                                     (lessp  pa  (difference  bd
3)))))
                    (! picture arrangement < block design - 3!
!            and picture arrangement < object assembly - 3!
                                                              !
and picture completion < block design - 3!
                                                              !
anb picture completion < object assembly - 3!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary    (qualifier    !    none!)
schizophrenia)
                          (additional ! none!)))
        (rule 126
              (score (and (lessp cm 10)
                          (and (lessp pc (difference oa 3))
                               (and  (lessp pc  (difference  bd
3))
                                     (and  (lessp pa (difference
oa 3))
                                          (lessp pa  (difference
bd 3))))))
                    (!  picture  arrangement and completion  <
block design - 3!
                     !             and picture arrangement  and
completion < object assembly - 3!
                     !             and comprehension < 10!))
              (hypothesis (main (qualifier ! none!) psychosis)
                          (secondary  (qualifier  !  paranoid!)
schizophrenia)
                          (additional ! none!)))
        (rule 127
              (observe 4 6)
              (observe 9 1)
              (hypothesis (main (qualifier ! none!) psychosis)
```

```
                        (secondary (qualifier !
acute or incipient!)
                                   schizophrenia)
                        (additional ! none!)))
        (rule 128
              (observe 9 1)
              (observe 10 10)
              (hypothesis  (main  (qualifier !  none!)  stable-
borderline-state)
                        (secondary none)
                        (additional ! none!)))
        (rule 129
              (observe 9 2)
              (hypothesis (main (qualifier ! none!)
                              feelings-of-being-looked-at-or-
criticized)
                        (secondary none)
                        (additional ! none!)))
        (rule 130
              (observe 9 3)
              (hypothesis  (main  (qualifier ! none!)  organic-
brain-syndrome)
                        (secondary none)
                        (additional ! none!)))
        (rule 131
              (observe 9 4)
              (hypothesis  (main (qualifier !  none!)  organic-
brain-syndrome)
                        (secondary none)
                        (additional ! none!)))
        (rule 132
              (observe 9 5)
              (hypothesis  (main  (qualifier ! none!)  organic-
brain-syndrome)
                        (secondary none)
                        (additional ! none!)))
        (rule 133
              (observe 9 6)
              (hypothesis  (main (qualifier !  none!)  organic-
brain-syndrome)
                        (secondary none)
                        (additional ! none!)))
        (rule 134
              (observe 9 7)
              (hypothesis  (main  (qualifier ! none!)  organic-
brain-syndrome)
                        (secondary none)
                        (additional ! none!)))
        (rule 135
              (observe 9 8)
              (hypothesis  (main (qualifier !  none!)  organic-
brain-syndrome)
                        (secondary none)
                        (additional ! none!)))
        (rule 136
```

```
              (observe 9 9)
              (hypothesis  (main  (qualifier ! none!)  organic-
brain-syndrome)
                           (secondary none)
                           (additional ! none!)))
        (rule 137
              (observe 9 10)
              (hypothesis  (main (qualifier !  none!)  organic-
brain-syndrome)
                           (secondary none)
                           (additional ! none!)))
        (rule 138
              (observe 9 11)
              (hypothesis  (main  (qualifier ! none!)  organic-
brain-syndrome)
                           (secondary none)
                           (additional ! none!)))
        (rule 139
              (observe 2 8)
              (hypothesis (main (qualifier ! none!) psychosis)
                           (secondary none)
                           (additional ! none!)))
        (rule 140
              (observe 10 3)
              (hypothesis  (main (qualifier !  none!)  organic-
brain-syndrome)
                           (secondary none)
                           (additional ! none!)))
        (rule 141
              (observe 5 4)
              (observe 10 4)
              (hypothesis  (main  (qualifier ! none!)  organic-
brain-syndrome)
                           (secondary none)
                           (additional ! none!)))
        (rule 142
              (observe 10 5)
              (hypothesis  (main (qualifier !  none!)  organic-
brain-syndrome)
                           (secondary none)
                           (additional ! none!)))
        (rule 143
              (observe 10 6)
              (hypothesis (main (qualifier ! none!)
                                antisocial-personality-
disorder)
                           (secondary none)
                           (additional ! none!)))
        (rule 144
              (observe 10 7)
              (hypothesis (main (qualifier ! none!)
                                antisocial-personality-
disorder)
                           (secondary none)
                           (additional ! none!)))
```

```
        (rule 145
              (observe 10 8)
              (hypothesis (main (qualifier ! none!)
                                antisocial-personality-
disorder)
                          (secondary none)
                          (additional ! none!)))
        (rule 146
              (observe 10 9)
              (hypothesis (main (qualifier ! none!)
                                antisocial-personality-
disorder)
                          (secondary none)
                          (additional ! none!)))))
```

APPENDIX C, OBSERVATION & BACKGROUND MENUS

These menus are prescribed within Dave as a list of ten lists which contain the entries in the menus presented by Item 4 of the Evaluation Menu. The clinical psychologist reviews these menus and enters the item number of those which were true at the time of administration of the WAIS test. See Appendix K for the procedure provided for the user to change these menus. This data is located in a file called "bomenu". This listing contains extra carriage returns because of space requirements.

```
(setq bckg-obs-menu
     '(((2   1   "  Failure of easy similarities items with
                 subsequent passes")
        (2 2
           "  Failure of easy Picture Completion items with
                 subsequent passes")
         (2  3  "  Failure of easy information items with
                 subsequent passes")
         (2 4 " Inability to perform digit span backwards")
         (2 5 " DK responses to comprehension")
         (2  6 " Missing easy Arithmetic items with subsequent
                 passes")
         (2 7 " Ability to work effectively and appropriately")
         (2 8
            "   Cognitive disruption with strange, morbid,
                 otherworldly content")
         (2 9 " Clang associations on vocabulary")
         (2 10 " Clinical evidence of depression")
         (2 11 " Preoccupation with minutia"))
        ((3 1 " No schophrenic verbalizations")
         (3 2 " Flippancy")
         (3 3 " Overmeticulous verbalization")
         (3 4 " Impulsive guessing on whole test")
         (3 5 " Inertia")
         (3 6 " Wild guessing")
         (3 7 " Lack of qualification")
         (3 8
            "   Limited perceptual distortions on Picture
                 Arrangement and Picture"
            "    Completion")
         (3 9 " Frequently gave irrelevant details")
         (3   10  "  Impulsive guessing on Imformation and
                 Vocabulary")
         (3 11 " Moralizing comments on Comprehension")
         (3 12 " Inability to concentrate on Arithmetic"))
```

```
((4 1 " Good visual-motor coordination")
 (4  2  "  Confused object identification on picture
          completion")
 (4   3  " Failure on easy Information items with
          subsequent passes")
 (4 4 " Rapid responses")
 (4  5  " Orderly resopnses, possibly occasional odd
          verbalizations")
 (4 6 " Anxiety")
 (4 7 " Presence of verbal word salad")
 (4   8    " Schizophrenia certain, from previous
          evaluation")
 (4 9 " Psychosis certain, from previous evaluation")
 (4 10 " Neurosis certain, from previous evaluation")
 (4 11 " Lack of verbal fluency")
 (4 12 " Few peculiar verbalizations")
 (4 13 " Bland guessing")
 (4 14 " Perceptual vagueness"))
((5 1 " Self-criticality")
 (5  2  "  A few perceptual distortions on Picture
          Arrangement")
 (5  3  "  A few perceptual distortions on Picture
          Completion")
 (5   4  "  Missing easy Comprehension items with
          subsequent passes")
 (5 5 " No serious hearing impairment")
 (5 6 " Supercilious attitude")
 (5 7 " Lack of elaboration")
 (5 8 " Good visual-motor speed")
 (5 9 " A schemer")
 (5 10 " Restricted formal education")
 (5  11  "  History of anxiety attacks or  an  observed
          attack")
 (5 12 " Grossly impaired reality contact")
 (5 13 " Reality contact good"))
((6 1
    " Onset of presenting problem immediately  followed
          an extremely"
    "      stressful situation")
 (6  2  "  Response  to  a  crisis  with  depression  or
          anxiety")
 (6 3 " Auditory hallucinations")
 (6 4
    "  Dynamic processes,  rather than simple inability
          to maintain reality"
    "      contact, responsible for impairment")
 (6 5 " Good family relationships")
 (6 6 " Onset of problem late in life")
 (6 7 " Has taken appreciable quantities of psychedelic
          drugs")
 (6   8  "  History  of  psychosis  but  presently  in
          remission")
 (6 9 " Lacks consideration for other people's feelings
          and rights")
 (6 10 " History of violent behavior")
```

```
 (6  11  " Lives predominately for immediate gain or
        pleasure")
 (6 12 " Overt hostility toward the examiner"))
((7 1 " Challenged examiner's authority, at least half-
        seriously")
 (7 2 " Overly active conscience")
 (7 3 " Normally active conscience")
 (7 4 " Specific gross inabilities present")
 (7  5 " Had trouble attaching names to objects during
        examination")
 (7 6
    " Had difficulty finding words to express ideas, as
        opposed to lack"
    "     of ideas")
 (7  7 " Signs of aphasia, other than two immediately
        preceding signs")
 (7 8 " Perseveration, motor or ideational or both")
 (7   9   " Rotations on Block Design even after
        demonstration")
 (7 10
    " Fluctuation ability ot retrieve facts in memory,
        such as own name")
 (7 11 " Severe motor handicaps")
 (7 12 " Slow motor speed")
 (7 13
    "  No sign of obvious organic insult in a thorough
        and reliable history")
 (7 14 " Global impression of mental retardation"))
((8 1
    "  Lack of or incorrect responses due to absence of
        information, rather"
    "      than bizarreness")
 (8 2 " Obvious presence of Down's Syndrome")
 (8 3 " Prior evaluation concluded mental retardation")
 (8  4  " Showed at least some lack of acceptance of
        failure")
 (8 5 " Completed at least one year of college")
 (8 6 " Showed extraordinary motor skill")
 (8  7 " Impression of higher ability than immediately
        apparent")
 (8 8 " Prior high occupational achievement")
 (8 9 " Prior high scholastic achievement")
 (8  10  " Extraordinary quality of definitions on
        Vocabulary")
 (8 11 " Ingenious problem-solving methods")
 (8  12  " Very active curiosity and fascination with
        surroundings")
 (8  13 " Made good guesses or approximations on failed
        items")
 (8 14 " If in school teacher endorses as gifted"))
((9 1
    "   Blurring of boundaries between separate
        categories, thoughts,"
    "     feelings, or experiences")
 (9 2
```

```
    "  Introduction of eyes where none exist,  or
       emphasis on existing eyes")
  (9 3 " Difficulty with abstract concept formation")
  (9 4 " Difficulty with hypothetical thinking")
  (9 5
    " Difficulty in keeping track of a train of thought
       or progress on a "
    "     task")
  (9 6
    "  Recognition of poor responses with desire to
       change them"
    "     but unable to make needed changes")
  (9 7 " Figure-ground reversals")
  (9 8
    " Confused directionality,  such as up-down,  left-
       right, east-west, etc.")
  (9 9 " Confabulation")
  (9 10
    "  Instances where the way to do something was
       obviously known"
    "       but could not put that knowledge into
       action")
  (9 11
    "   Difficulties did not reflect  psychological
       embellishment and"
    "     thematic richness"))
((10 1
     " Cognitive disruption,  with obvious struggle to
       form rather mundane"
     "     thoughts or concepts")
  (10   2  "  Withdrawal from conventional modes  of
       experience")
  (10  3  " Particular difficulty with Comprehension's
       proverbs")
  (10 4 " Frequent misinterpretation of questions")
  (10 5 " Understood what was expected but difficulty in
       execution")
  (10 6 " Polite, measured tone of interaction")
  (10    7   "   Impression   of   hyperalertness   and
       watchfulness")
  (10 8
     "  Mistrust  of examiner and not under  evaluation
       for criminal"
     "     proceedings")
  (10 9 " Felt put upon if put under even mild  pressure
       or when failing")
  (10 10 " Little or no anxiety"))
((11 1 " Bizzare and remote associations"))))
```

APPENDIX D, SD-QUALIFIERS

This is the set of rules which mark a clinical entity for superqualification or disqualification. If a clinical entity is found to have its evidence present for disqualification then it is no longer considered. If it is found to have its evidence present for superqualification then the comments are placed on the output report for that clinical entity and it receives 100 points on its observation evidence score. T

This rule is located in a file called "sd-qualifiers". The procedure for the user to change this rule is located in Appendix S and the code in Appendix I.

```
(setq sd-qualifiers
      '((neurosis (disqualifiers (and (observe 5 12)
                                      (observe 6 1)))
                  (superqualifiers (and (observe 5 11)
                                        (observe 5 13)
                                        (observe 6 2))
           ("observe: History of anxiety attacks or an observed
attack"
            "and  observe: Reality contact good"
            "and observe:  Response to a crisis with depression
or anxiety")))
        (psychosis (disqualifiers
                    (and (observe 5 13)
                         (observe 6 4)
                         (observe 6 6)
                         (not (observe 6 8))
                         (not (observe 6 2))
                         (or (observe 8 8) (observe 8 9))))
                   (superqualifiers (and (observe 6 3)
                                         (observe 6 4))
         ("observe: Auditory hallucinations"
          "and observe:  Dynamic processes,  rather than simple
inability to maintain reality"
           "             contact, responsible for impairment")))
        (antisocial-personality-disorder (disqualidiers
                                          (or  (observe  7   2)
                                               (observe 7 3)))
                                         (superqualifiers  (and
                                                 (observe 5 13)
                                                 (observe 6 9)
```

```
                                                    (observe 6 10))
                        (" observe: Reality contact good"
                         "and  observe:  Lacks  consideration  for
                                other people's feelings and rights"
                         "and    observe:    History    of    violent
                                behavior.")))
          (organic-brain-syndrome (disqualifiers
                                    (and (observe 4 7)
                                         (observe 6 4)
                                         (observe 7 13)
                                         (observe 11 1)))
                                  (superqualifiers (and (observe 7 4)
                                                        (or (observe 7 5)
                                                            (observe 7 6)
                                                            (observe 7 7)))
              ("observe: Specific gross inabilities present"
               "and     observe:  Had trouble attaching names  to
objects during examination"
               "      or observe:  Had difficulty finding words to
express ideas, as opposed to lack of ideas"
               "       or observe:   Signs of aphasia,  other than
                                two immediately preceding signs.")))
          (mental-retardation (disqualifiers
                                (or (greaterp fsiq 69)
                                    (and (lessp fsiq 70)
                                         (observe 7 14)
                                         (observe 8 7))))
                              (superqualifiers (and (lessp fsiq 70)
                                                    (observe 7 14)
                                                    (observe 8 2))
             ("score: full scale IQ ( 70"
              "and   observe:    Global   impression   of   mental
retardation"
              "and   observe:    Obvious   presence   of   Down's
Syndrome")))
          (giftedness (disqualifiers nil)
                      (superqualifiers (and (greaterp fsiq 129)
                                            (or  (observe  8   7)
                                                 (observe 8 9)))
              ("score: full scale IQ ) 129"
               "and    observe: Impression of higher ability than
immediately apparent"
              "or observe: Prior high scolastic achievement")))))
```

APPENDIX E, CLINICAL ENTITIES

This is the list of main clinical entities which are being considered by Dave. The rules may contain other conditions such as depression or schizophrenia but they must be in this list before they will receive a separate evaluation on the output report.

The procedure for the user to add to or delete items from this list is contained in Appendix M. This data is located in a file called "ce".

```
(setq clinical-entities
                '(neurosis
                  psychosis
                  antisocial-personality-disorder
                  organic-brain-syndrome
                  mental-retardation
                  giftedness))
```

APPENDIX F, MENU HANDLER

This file called "menu" contains the code which controls Dave's two main menus. It also contains a list of all the WAIS score names used in Dave and the routine to accept their values from the user and calculate the composite scores. It also contains a number of functions which are used repeatedly throughout other files to format output, initialize variables, and read a line of text from the screen as string.

This is a list of the entries which will be printed on the terminal when the initial menu is executed.

```
(setq first-menu '("            EVALUATION MENU"    " "
        "1. Enter previous WAIS scores."
       "2. Enter current WAIS scores."
       "3. Enter local scores."
        "4. Enter observation & background information."
       "5. Perform evaluation of test results."
        "6. View latest report on terminal."
        "7. Print latest report."
        "8. Produce report with all rules true."
        "9. Teach Dave Menu."
        "10. Exit Dave."]
```

This is a list of the entries which will be printed on the terminal when Item 9 of the Evaluation Menu is executed.

```
(setq modify-dave-menu '("          TEACH DAVE MENU" " "
        "1. Change observation & background menus."
        "2. Change the clinical entities to be examined."
        "3. Change the local score names to be examined."
        "4. Change sign rules."
        "5. Change Superqualifier/Disqualifier rules."
        "6. Make file changes permanent."
        "7. Return to main evaluation menu."
        "8. Exit Dave."]
```

This is the procedure which is called to start Dave. It

prints the Evaluation Menu and calls the functions as requested by the user entering menu item numbers.

```
(defun dave ()
   (blank-line 3)
   (cond (first (load 'ce)
                (load 'ls)
                (load 'bomenu)
                (load 'sd-qualifiers)
                (load 'rules)
                (mapcar 'setzero-a (mapcar 'car all-prev-score-
list))
                (mapcar 'setzero-a (mapcar 'car local-score-
list))
                (mapcar 'setzero-a (mapcar 'car all-score-
list))
                (setq obs-list nil)
                (setq first nil)))
   (mapcar 'printlit2 first-menu)
   (caseq (read)
        (1 (get-scores prev-score-list)
           (calc-prev-composites)
           (dave))
        (2 (get-scores score-list)
           (calc-composites)
           (dave))
        (3 (get-scores local-score-list)
           (dave))
        (4 (do-obs-menus)
           (dave))
        (5 (do-sd-qualifiers)
           (mapcar 'rule sign-rules)
           (produce-report)
           (dave))
        (6 (exec "more report")
           (dave))
        (7 (msg N N N "Do you want to print the latest report?"
N)
           (msg "Enter y/n. ")
           (cond ((eq (read) 'y)(exec "lpr report")))
           (dave))
        (8 (setq fired-clinical-entities clinical-entities)
           (init-fired-clinical-entities)
           (mapcar 'all-rules sign-rules)
           (produce-totally-fired-report)
           (dave))
        (9 (do-modify-dave-menu))
        (10 (msg N N "Goodbye from Dave" N N))
        (help (exec "more help1")
           (dave))
        (t (msg N N "Please enter a valid menu item number." N)
           (dave]
```

This is the procedure which is called by Item 9 of the Evaluation Menu. It prints the Teach Dave Menu and calls the procedures called for be the user by entering menu item numbers.

```
(defun do-modify-dave-menu ()
   (blank-line 3)
   (mapcar 'printlit2 modify-dave-menu)
   (caseq (read)
        (1 (load 'chobs)
           (init-obs-menu m-nums bckg-obs-menu)
           (change-observation-menus)
           (do-modify-dave-menu))
        (2 (load 'chce)
           (change-clinical-entities)
           (do-modify-dave-menu))
        (3 (load 'chce)
           (change-local-score-list)
           (do-modify-dave-menu))
        (4 (load 'chrule)
           (change-rule)
           (do-modify-dave-menu))
        (5 (load 'chrule)
           (load 'chsdq)
           (change-sd-qualifier)
           (do-modify-dave-menu))
        (6 (load 'permfile)
           (write-permfile)
           (do-modify-dave-menu))
        (7 (dave))
        (8 (msg N N "Goodbye from Dave." N N))
        (help (exec "more help9")
           (do-modify-dave-menu))
        (t  (msg N "ERROR!!!!   Please enter a valid menu  item
number." N)
           (do-modify-dave-menu]
```

This is the list of all of the current WAIS score identifier names and their descriptions.

```
(setq score-list '((i information)
                   (s similarities)
                   (a arithmetic)
                   (v vocabulary)
                   (cm comprehension)
                   (ds " digit span")
                   (pc " picture completion")
                   (pa " picture arrangement")
                   (bd " block design")
```

```
                    (oa " object assembly")
                    (cd " digit symbol")
                    (viq " verbal IQ")
                    (piq " performance IQ")
                    (fsiq " full scale IQ")
                    (df " digits forward")
                    (db " digits backward"]
```

This is a list of all of the scores which are calculated based upon the current WAIS scores above.

```
(setq all-score-list (append score-list '(
                    (vs " verbal scatter")
                    (ps " performance scatter")
                    (vmean " verbal mean")
                    (pmean " performance mean")]
```

This is a list of previous WAIS score identifier names and their descriptions.

```
(setq prev-score-list '((prev-i " previous information")
                    (prev-s " previous similarities")
                    (prev-a " previous arithmetic" )
                    (prev-v " previous vocabulary")
                    (prev-cm " previous comprehension")
                    (prev-ds " previous digit span")
                    (prev-pc " previous picture completion")
                    (prev-pa " previous picture arrangement")
                    (prev-bd " previous block design")
                    (prev-oa " previous object assembly")
                    (prev-cd " previous digit symbol")
                    (prev-viq " previous verbal IQ")
                    (prev-piq " previous performance IQ")
                    (prev-fsiq " previous full scale IQ")
                    (prev-df " previous digits forward")
                    (prev-db " previous digits backward"]
```

This is a list of all scores which are calculated based on the values of the previous WAIS scores above.

```
(setq all-prev-score-list (append prev-score-list '(
                    (prev-vs " previous verbal scatter")
                    (prev-ps " previous performance scatter")
                    (prev-vmean " previous verbal mean")
                    (prev-pmean " previous performance mean"]


          (t (terpr) (blank-line (sub1 x]
```

This is a utility function used to set an identifier to nil.

```
(defun setnil (x)
    (set x 'nil]
```

This is a utility function used to set an identifier to zero.

```
(defun setzero-a (score)
   (set score 0]
```

This is a utility function used to print a blank line x number of times.

```
(defun  blank-line (x)
   (cond ((zerop x) t)
```

The following printlit functions are used throughout Dave to format output in various ways.

```
(defun printlit (l)
    (cond ((null l) t)
          (t  (print (caar l))(princ (cadar l))
              (terpr)
              (printlit (cdr l]

(defun printlit1 (l)
    (cond ((null l))
          (t (print (cadar l))
             (mapcar 'printlit2 (cddar l))
             (printlit1 (cdr l)]

(defun printlit1a (l)
    (cond ((null l))
          (t (msg  (caar l) " " (cadar l))
             (mapcar 'printlit2 (cddar l))
             (printlit1a (cdr l)]
```

```
(defun printlit2 (l)
    (princ l)
    (terpr]

(defun printlit2a (l)
    (princ "      ")
    (princ l)
    (terpr]

(defun printlit3 (l)
    (cond ((null l))
          (t (princ (car l))
             (terpr)
             (printlit3 (cdr l]

(defun printlit4 (l)
   (msg (car l) " = " (eval (car l)) "      " (cadr l) N)]
```

This is called by the function dave. It is used to obtain the value of local or WAIS scores from the terminal user and assign them to identifiers in l. l will have the value of one of the score lists above.

```
(defun get-scores (l)
   (cond ((null l)(msg N "    There are no local scores listed"
N))
   (t (mapcar 'get-score l]
```

This is used by get-scores above to perform its task for one score. At this time, l will have the value of one of the identifiers and its description, eg. (i information).

```
(defun get-score (l)
   (msg  "Enter score for " B )
   (princ (cadr l))
   (princ '"    ")
   (set (car l) (read]
```

This is used to give first a value when the program is first started. First is used in the function dave to determine if loading of rules, meuns, and local score lists, and initialization of score identifiers and obs-list is necessary.

```
(setq first t)
```

This is used to calculate the composite scores for the current WAIS scores.

```
(defun calc-composites ()
   (setq vs (- (max i s a v cm ds) (min i s a v cm ds)))
   (setq ps (- (max pc pa bd oa cd)(min pc pa bd oa cd)))
   (setq vmean (div-round (plus i s a v cm ds) 6))
   (setq pmean (div-round (plus pc pa bd oa cd) 5]
```

This is used to calculate composite scores for the previous WAIS scores.

```
(defun calc-prev-composites ()
   (setq  prev-vs  (- (max prev-i prev-s prev-a prev-v  prev-cm
prev-ds)
                      (min  prev-i prev-s prev-a  prev-v  prev-cm
prev-ds)))
   (setq  prev-ps (- (max prev-pc prev-pa prev-bd prev-oa prev-
cd)
                      (min prev-pc prev-pa prev-bd prev-oa  prev-
cd)))
   (setq prev-vmean (div-round
                  (plus  prev-i  prev-s  prev-a  prev-v  prev-cm
prev-ds)
                  6))
   (setq prev-pmean (div-round
                  (plus prev-pc prev-pa prev-bd prev-oa prev-cd)
                  5]
```

This is used by the two functions above to do division with rounded answers to the nearest whole number.

```
(defun div-round (x y)
   (setq l (Divide x y))
   (cond ((lessp (* 2 (cadr l)) y)(car l))
         (t (add1 (car l]
```

This function is used throughout Dave to read a line of text and use it in the form of a string. One space is padded to the front of the string.

```
(defun readln ()
   (readc)  ;reads the preceeding line feed and discards
   (let ((item " "))
      (readln-help)

(defun  readln-help  ()  ;reads the remaining characters and
                         ;concats them together
   (let ((atom (readc)))
      (setq item (concat item atom))
      (cond ((eq 10 (tyipeek)) item)
            (t (readln-help)
```

APPENDIX G, EVALUATE SIGN-RULES

This file contains the functions which test the sign-rules to determine if the evidence is present and manipulates the lists to add the appropriate information to the output lists. It is a major part of the evaluation executed from Item 5 of the Evaluation Menu. It also contains several functions to initialize identifiers and a modification of the first two functions to produce the report from Item 8 of the Evaluation Menu, which has all rules with their evidence present.

This is the initial function to evaluate the sign-rules. Points is used as a switch to control the scoring process. When the first hypothesis of a rule causes an increase in one of the scores then first is set to nil. (Only the first hypothesis which contains one of the main clinical entities is allowed to be scored). Fired-rule is the list of the evidence which is found to be present in a rule. It is used along with each hypothesis for the output report. At this point l contains one complete sign-rule.

```
(defun rule (l)
   (setq points t)
   (setq fired-rule nil)
   (check-fire (cddr l]
```

This function breaks apart the sign-rule into its parts and acts upon them appropriately. R is initially the entire sign-rule without the word rule and the rule number. If any of the evidence is not present then the evaluation stops. If it is all true then the certain or the hypothesis sections will place

the fired-rule list on the appropriate output list.

```
(defun check-fire (r)
   (caseq (caar r)
        (hypothesis (mapcar 'add-hyp r))
        (score
                  (cond ((eval(cadar r))
                              (setq fired-rule (cons
                                                   (append

'(score)(caddar r))
                                                   fired-rule))
                              (check-fire (cdr r)))))
        (bckgrnd
                  (cond ((eval (car r))
                              (setq  fired-rule  (cons  (car  r)

fired-rule))
                              (check-fire (cdr r)))))
        (observe
                  (cond ((eval (car r))
                              (setq  fired-rule  (cons  (car  r)

fired-rule))
                              (check-fire (cdr r)))))
        (not      (cond ((eval (car r))
                              (setq  fired-rule  (cons  (car  r)

fired-rule))
                              (check-fire (cdr r)))))
        (certain
                 (add-certain-hyp (cadr r)(cadar r)))
        (t (msg N "error in check-fire" N B B " r is " r])
```

This is the function which adds the fired-rule list, which contains the rule's evidence, and the hypothesis information to the output list of the entity which is contained in the last portion of the main hypothesis clause. At this point, h contains one of the hypothesis causes. If points is still true then it will give a point to the clinical entity for the type of evidence contained in the first piece of evidence contained in the rule and set points to nil.

```
(defun add-hyp (h)
   (let ((ce (car (last (cadr h)))))
```

```
      (cond ((memq ce fired-clinical-entities)
                    (set  ce (cons (append (cdr h) fired-rule)
(eval ce)))
                    (cond ((eval points)
                      (cond ((eq 'not (caar(last fired-rule)))
                             (let ((id (implode
                                         (append
                                            (explode ce)
                                            (explode
                                                 (caadar(last
fired-rule)))))))
                                        (set id (add1 (eval id)))
                                        (setq points nil)))
                          (t (let ((id (implode
                                         (append
                                            (explode ce)
                                            (explode(caar(last
fired-rule)))))))
                                       (set id (add1 (eval id)))
                                       (setq points nil))   )))))
         (t (setq other-fired-rules
                  (cons
                      (append (cdr h) fired-rule)
                      other-fired-rules]
```

This is the function which deals with rules with a might-add flag. If the evidence preceding this clause was present then the fired-rule list and the hypothesis information which follows will be added to the might-add list for the clinical entity contained in the certain clause.

```
(defun add-certain-hyp (h ce)
   (let ((ce-ma (implode (append
                              (explode ce)
                              (explode 'might-add)))))
       (set ce-ma (cons
                    (cons (cadr h) fired-rule)
                    (eval ce-ma]
```

The next two functions are utility functions used to

initialize the evidence scores, the output lists, and the might-add lists.

```
(defun init-fired-clinical-entities ()
   (mapcar 'setzero fired-clinical-entities)
   (mapcar 'setnil fired-clinical-entities)
   (setnil 'other-fired-rules]

(defun setzero (e)
   (set (implode(append(explode e)(explode 'score))) 0)
   (set (implode(append(explode e)(explode 'observe))) 0)
   (set (implode(append(explode e)(explode 'bckgrnd))) 0)
   (set (implode(append(explode e)(explode 'might-add))) nil]
```

These are modifications of rules and check-fire without the check to see if the evidence is present. The result is that a report is produced which has all of the rules listed with their clinical entity.

```
(defun all-rules (l)
   (setq points t)
   (setq fired-rule nil)
   (fire-total-report (cddr l]

(defun fire-total-report (r)
   (caseq (caar r)
       (hypothesis (mapcar 'add-hyp r))
       (score (setq fired-rule (cons
                                   (append '(score)(caddar r))
                                   fired-rule))
                 (fire-total-report (cdr r)))
       (bckgrnd (setq fired-rule (cons (car r) fired-rule))
                 (fire-total-report (cdr r)))
       (observe (setq fired-rule (cons (car r) fired-rule))
                 (fire-total-report (cdr r)))
       (not (setq fired-rule (cons (car r) fired-rule))
                 (fire-total-report (cdr r)))
       (certain (add-certain-hyp (cadr r)(cadar r)))))
```

APPENDIX H, EVALUATE SD-QUALIFIERS

The file called "sdq" contains functions which evaluate the sd-qualifiers in much the same way that "driver" did for the sign-rules. It also contains the functions to evaluate the observe and background evidence used here and in the sign-rules.

These are the functions which are used to see if a particular menu number and item number are contained on the obs-list which is a list of all of the selected menu items. There is essentially no difference in the way that these functions treat observation and background information. They simply return t or nil based upon their presence on obs-list.

```
(defun observe (menu sel)
   (match-obs menu sel obs-list]

(defun bckgrnd (menu sel)
    (match-obs menu sel obs-list]

(defun match-obs (menu sel l)
   (cond ((null l) nil)
         ((and (eq menu (caar l))
               (eq sel  (cadar l))))
          (t (match-obs menu sel (cdr l]
```

This function initiates the evaluation of the sd-qualifier rules. It initializes its major identifiers and calls the evaluation function with each of the rules in the sd-qualifier list.

```
(defun do-sd-qualifiers ()
   (setq fired-clinical-entities clinical-entities)
   (init-fired-clinical-entities)
```

```
(mapcar 'fire-sd-qualifier sd-qualifiers]
```

This is the function which evaluates each rule. If the evidence is found to be true for the disqualification clause then it is removed from the fired-clinical-entities list and no longer considered. If the evidence is found to be true for the superqualification then the word "superqualification" and the description of the superqualification evidence is placed on the output list for that clinical entity and it is given an observation score of 100.

```
(defun fire-sd-qualifier (l)
   (cond  ((eval (cadadr l))
                  (setq fired-clinical-entities
                        (remove    (car   l)    fired-clinical-
entities)) )
          ((eval (cadaddr l))
                  (set (car l) (cons
                                  (list  (cons  'superqualified
(caddaddr l)))
                                  (eval (car l))))
                  (let       ((n      (implode(append(explode(car
l))(explode 'observe)))))
                      (set n (+ 100 (eval n]
```

APPENDIX I, CHANGE SD-QUALIFIERS

This file, called "chsdq", contains functions which allow the user to change the sd-qualifiers, which disqualify or superqualify clinical entities. It is executed by using Item 5 of the Teach Dave Menu. They initially present another menu which allows the user to choose to add , change, display permanent rules, or the latest changes which he has made.

This is the function which presents the menu which allows the user to choose between adding, changing, or viewing a rule. When changes are made using these functions, they are only made in memory. In order for the change to be recorded permanently Item 6 of the Teach Dave Menu must be used. This is why there is a distinction between Item 1 and 2 on this menu.

```
(defun change-sd-qualifier ()
   (msg   N  N  N  "**************************   SD-QUALIFIERS
************************" N N)
   (msg "1. View latest permanent sd-qualifiers on screen" N)
   (msg "2. View current sd-qualifiers on screen." N)
   (msg "3. Search sd-qualifiers for possible changes " N)
   (msg "4. Add new sd-qualifier" N)
   (msg "5. Exit" N)
   (msg "       Enter number of selection. " N)
   (caseq (read)
       (1 (exec "more sd-qualifiers"))
       (2 (pp-form sd-qualifiers))
       (3 (get-inside-sdq-list))
       (4 (add-sd-qualifier))
       (5 nil)
       (help (exec "more help25a")(change-sd-qualifier))
       (t (msg N "ERROR!!!! Enter 1 thru 4. Try again." N N N)
          (change-sd-qualifier))]
```

This function is used to pick out one on the rules within the sd-qualifiers so that it may be changed. Then it must put the rules back together.

```
(defun get-inside-sdq-list ()
   (do ((front nil (cons target front))
        (target (car sd-qualifiers) (car end))
        (end (cdr sd-qualifiers) (cdr end)))
       ((or (null target)
            (setq new-sd-qualifier (check-sdq-target target)))
            (cond ((eq new-sd-qualifier '+)
                                    (msg     N     "      Abandoned
substitution" N ))
                          (t    (setq   sd-qualifiers    (append
(reverse front)
                                                          (list new-
sd-qualifier)
                                                          end)]
```

This function is used by the one above to ask the terminal user if the target rule is the one which is to be changed. Target and 1 are the same at this point.

```
(defun check-sdq-target (1)
   (msg N N N N "Is this the one to change?" N (pp-form target)
N )
   (msg  "           Enter  y to substitute for  this  clinical
entity" N)
   (msg "                  n to get next clinical entity" N)
   (msg "                  + to quit search " N)
   (let ((answer (read)))
       (cond ((eq answer '+)'+)
             ((eq answer 'help)(exec "more help25b')(check-sdq-
target 1))
             ((eq answer 'y)
              (check-substitute-sd-qualifier]
```

This function calls the next to get the new sd-qualifier and then presents it to the user to see if it is correct. The user has the option of making the substitution in memory,

re-entering the new rule, or exiting without substitution.

```
(defun check-substitute-sd-qualifier ()
   (let  ((new-sd-qualifier  (get-substitute-sd-qualifier  (car
target))))
       (msg  N "Is this new sd-qualifier correct?"  N  (pp-form
new-sd-qualifier) N )
       (msg "          Enter y to record substitution" N)
       (msg "                n to redo rule" N)
       (msg "                + to abandon the substitution. " N)
          (let ((answer (read)))
             (cond ((eq answer 'y) new-sd-qualifier)
                   ((eq answer '+) '+)
                   ((eq answer 'help)(exec "more help25c")
                                        (check-substitute-sd-
qualifier))
                   (t (check-substitute-sd-qualifier)]
```

This function presents explanatory information on the terminal and calls the functions to get the two parts to the new sd-qualifier. Then this function puts the key words together with the two new parts to form the new sd-qualifier for the clinical entity specifier in ce.

```
(defun get-substitute-sd-qualifier (ce)
   (blank-line 5)
   (msg  "                  GET REPLACEMENT SD-QUALIFIER RULE"  N
N)
   (msg   "You   will   be selecting two   patterns   of   evidence
which" N)
   (msg  "will disqualify a clinical entity from  consideration
or" N)
   (msg "give it superqualification,  ie indicate a very strong
" N)
   (msg  "probability  of being present.  An  english  language
description" N)
   (msg "of the superqualifying evidence be asked for next." N)
   (list ce (list 'disqualifiers (get-disqualifier))
            (cons 'superqualifiers (get-superqualifier)]
```

This function is used when the user wants to add a sd-qualifier rule to a new clinical entity which does not already have an sd-qualifier rule. The new clinical entity must have already been added to the clinical entity list using Item 2 of the Teach Dave Menu. It must not have a current sd-qualifier rule for this entity. If these conditions are met, then it calls the functions which get the two parts of the new rule and puts the new parts together with the proper key words into an new rule. Then it adds the new rule onto the end of the current sd-qualifers.

```
(defun add-sd-qualifier ()
   (blank-line 10)
   (msg "                          ADD SD-QUALIFIER" N N)
   (msg "You will be asked for a clinical entity which has been
added to the " N)
   (msg "clinical entities list but does not have a current sd-
qualifier rule." N)
   (msg  "Then you must enter evidence for disqualification and
superqualification." N)
   (msg  "You  then  will  be asked  for  an  English  language
description of " N)
   (msg "the superqualification evidence." N N)
   (msg "Now enter the new clinical entity." N)
   (let ((ce (read)))
      (cond   ((eq  ce  'help)(exec   "more   help25d")(add-sd-
qualifier))
            ((not (member ce clinical-entities))
               (msg "ERROR!!!!  Clinical entity not on current
list." N))
            (t (cond ((is-ce-in-sdq sd-qualifiers)
                      (t (msg "ERROR!!!!"
                           "   There  is  already  an   sd-
qualifier"
                           " for that entity." N)))
                     (t (setq sd-qualifiers
                              (cons
                                  (list                    ce
                                        (list 'disqualifiers
                                              (get-
disqualifier))
                                        (cons
'superqualifiers
                                              (get-
superqualifier)))
                                  sd-qualifiers]
```

This is the function used by add-sd-qualifier to find out if the clinical entity, ce, is contained in the current sd-qualifiers. It returns t or nil.

```
(defun is-ce-in-sdq (l)
   (cond ((null l) nil)
         ((eq ce (caar l)))
          (t (is-ce-in-sdq (cdr l)]
```

The next two functions are the ones used to get the disqualifier and superqualifier clauses of the sd-qualifier rule. They present the title, call other functions to get evidence *and* descriptions of the superqualification evidence, and put them together if necessary.

```
(defun get-disqualifier ()
   (msg N N N "********* GET EVIDENCE FOR DISQUALIFICATION OF "
ce "***************" N N)
   (get-sdq-evidence]

(defun get-superqualifier ()
   (msg  N N N "********** GET EVIDENCE FOR  SUPERQUALIFICATION
OF " ce "************" N N )
   (list (get-sdq-evidence)
         (get-score-description)]
```

This is the function which assembles an evidence rule based on a series of menu entries by the user. It uses functions which are located in the "chrule" file. See Appendix S for a detailed description of its functions. It differs from the functions which form sign-rules in that this allows combination on WAIS score comparison, observation, and background evidence checks using AND and OR as binary operators. The sign-rules

assume that all evidence listed must be present, ie AND only.

```
(defun get-sdq-evidence ()
   (msg N "                    Type of Evidence")
   (msg N "Now choose one of the score comparisons or checks of
Observations" )
   (msg  N "and Background Information as evidence.   If two of
these are going ")
   (msg N "to be linked together with AND or OR,  then  precede
the phrase with ")
   (msg  N "a left parenthesis.  Right parentheses are done for
you. Do not put")
   (msg N "parentheses around single items of evidence (2  thru
6 below). Here ")
   (msg N "is a rather abreviated example.")
   (msg  N "((Observation-2 AND Observation-3) OR  (WAIS-Score-
Compare AND Background))" N N)
   (msg N "1. (  Left parenthesis, begins new AND/OR phrase.")
   (msg N "2. WAIS Score Comparison.")
   (msg N "3. Observation.")
   (msg N "4. Background.")
   (msg N "5. Absence of Observation.")
   (msg N "6. Absence of Background Indicator")
   (msg N "       Enter menu item number.")
   (caseq (read)
      (1 (get-sdq-logical-operator))
      ((2 W w) (get-relative-operator))
      ((3 O o) (get-observe-evidence))
      ((4 B b) (get-bckgrnd-evidence))
      (5 (list 'not (get-observe-evidence)))
      (6 (list 'not (get-bckgrnd-evidence)))
      (help (exec "more help25f")(get-sdq-evidence))
      (t (msg N "ERROR!!!! Enter 1 thru 5. Try again." N)
         (get-sdq-evidence)]
```

The next two functions are used by get-sdq-evidence whenever a user indicates that he has a left parenthesis which marks the begining of a new AND/OR clause. They present the option of using AND or OR, and call get-sdq-evidence recursively for another piece of evidence. Both logical operators are restricted to only two operators because most people think of them as being binary and infix.

```
(defun get-sdq-logical-operator ()
   (let ((first (get-sdq-evidence)))
      (sdq-logical-operator-menu)]
```

```
(defun sdq-logical-operator-menu ()
   (msg N "              LOGICAL OPERATORS" N)
   (msg N "1. AND ")
   (msg N "2. OR")
   (msg N "    Enter menu item number.")
   (caseq (read)
    ((1 AND and a)(list 'and (get-sdq-evidence) first))
    ((2 OR or o)(list 'or (get-sdq-evidence) first))
    (help (exec "more help25e)(sdq-logical-operator-menu))
    (t (msg N "ERROR!!!! Enter 1 or 2. Try again." N)
       (sdq-logical-operator-menu]
```

APPENDIX J, OBSERVATION & BACKGROUND MENU EXECUTION

This file, called "obsmenu", contains the functions which allow the user to enter responses regarding which one of the menu items are true for the examinee.

This function calls another function to present the menus and retreive the responses. Then this function presents the list of resopnses to the user. If the user is not happy with the correctness of this list then he should execute Item 4 of the Evaluation Menu again which will cause this procedure to execute once more.

```
(defun do-obs-menus ()
   (setq obs-list nil)
   (mapcar 'do-one-obs-menu bckg-obs-menu)
   (msg N N "    Here are all your selections" N)
   (printlit1a obs-list]
```

This function presents one of the menus which assigned to the identifier 1. It calls make-menu-selections to retrieve the entries from the terminal user. This function then presents this menu's selections to the user. If he has selected any of the entries from this menu, he has the option of making the entries over again or accepting them. When the entries are accepted, they are added to obs-list.

```
(defun do-one-obs-menu (1)
   (setq obs-menu-list nil)
   (blank-line 15)
   (msg "     Observations & Background Menu #" (caar 1) N N)
   (printlit1 1)
   (make-menu-selections 1)
   (cond ((null obs-menu-list))
```

```
          (t
             (msg N N "These are your selections" N N )
             (printlit1 obs-menu-list)
             (msg N "     Are these correct?  (enter y or n)" B
B)
             (let ((answer (read)))
                  (cond ((eq answer 'help)(exec "more help14")
                                          (do-one-obs-menu 1))
                        ((eq answer 'n)(do-one-obs-menu 1))
                        (t (setq obs-list (append obs-list obs-
menu-list]
```

This function is used by the one above to obtain the entries from the terminal user. If the entry is a "+" then it stops, otherwise it sends the item read to match-list.

```
(defun make-menu-selections (l)
   (msg N "Enter the number of selection,  or + when complete."
B B)
   (let ((item (read)))
      (cond ((eq item 'help)(exec "more help14")
                            (do-one-obs-menu 1))
            ((eq item '+))
            (t (match-list item l)(make-menu-selections l]
```

This function is used by the one above to check to make sure that the entry made by the terminal operator is valid. It does this by calling match-item with each of the menu entries in l until a match is found or the end of the list is found.

```
(defun match-list (item l)
    (cond
         ((null l) (msg "Incorrect response" N))
         ((match-item item (car l)))
          (t (match-list item (cdr l]
```

This function is used by the one above to find out if a particular item number "item" is valid for a particular menu "l". If it is valid then it is added to obs-menu-list which is

the list of valid entries for this menu.

```
(defun match-item (item l)
     (cond  ((equal item (cadr l))(setq obs-menu-list  (cons  l
obs-menu-list]
```

APPENDIX K, CHANGE OBSERVATION & BACKGROUND MENU

This file, called "chobs", contains functions which allow the user to add to or change the wording on the Observation & Background Menus.

This function presents the user with yet another menu allowing him to choose between additional menu items and changing the current ones. It then calls the appropriate function.

```
 (defun change-observation-menus ()
   (msg  N "* * * * * * * * * * CHANGE OBSERVATION  &  BACKGROUND
MENUS * * * * * * * * *" N N)
   (msg "1. Add a new menu item." N)
   (msg "2. Change wording on a menu item." N)
   (msg "3. Exit." N)
   (msg "       Enter menu item number." B)
   (caseq (read)
      ((1 a A add)(add-menu-item))
      ((2 c C change)(change-menu-item))
      (3 nil)
      (help (exec "more help21a")
            (change-observation-menus))
      (t (msg N "ERROR!!!! Enter 1 thru 3. Try again." N)
         (change-observation-menus)]
```

This list contains identifiers used by the functions which modify these menus. They are identifiers which evaluate to lists of item identifiers. The list associated with m2, for example, would be (m21 m22 m23 m24 m25 m26 m27 etc.). Then these identifiers are associated with the individual menu entries, such as (2 4 " Inability to perform digit span backwards"). Note: The capability to add additional menus has not been provided. If it becomes necessary, then changes must be made in three places to the code. Add m12 to this list. Add 12 to the liist in change-menu-item. Also, add a nil list

to the bckg-obs-menu.

```
(setq m-nums '(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11))
```

The next two functions are used to assign the current menu lists to the identifiers as described in the m-nums above.

```
(defun init-obs-menu (m-num m-list)
   (cond ((null m-num))
         (t (set-m-list (car m-num) (car m-list))
            (init-obs-menu (cdr m-num) (cdr m-list)]


(defun set-m-list (m-num m-list)
   (set m-num nil)
   (do ((l m-list (cdr l)))
      ((null l) (set m-num (reverse (eval m-num))))
      (let ((item (concat 'm (caar l) (cadar l))))
           (set item (car l))
           (set m-num (cons item (eval m-num]
```

This function is used to change the wording on one of the existing menu entries. They cannot be deleted completely although a blank line is acceptable for an entry. A "+" will exit the function and a "help" will display explanatory information and return to this function. An error message is given if the menu number is not valid. Otherwise, it will display the menu and ask for the item to be changed. If Item 3 of menu number 2 is chosen then a new value will be assigned to the identifier m23 as described in m-nums above. Save-bckg-obs is called to convert these lists of lists of identifiers into the new value of bckg-obs-menu.

```
(defun change-menu-item ()
   (msg  N "******** Change wording on observation & background
menus ********" N)
   (msg  "The menus are numbered 2 thrugh 11.  Which one  would
you like to change?" N)
   (msg "     Enter + when you are complete" N)
```

```
   (let ((menu (read)))
      (cond ((eq menu '+))
            ((eq  menu 'help)(exec "more help21b")(change-menu-
item))
            ((not (member menu '(2 3 4 5 6 7 8 9 10 11)))
                              (msg N " You made an  error.  Try
again" N)
                              (change-menu-item))
            (t (msg N  "     Here is menu number " menu N N)
               (printlit1 (nth (- menu 2) bckg-obs-menu))
               (msg N " Which item would you like to change?" B
B)
               (let ((item (read)))
                  (cond ((eq item '+))
                        ((eq     item     'help)(exec     "more
help21b")(change-menu-item))
                        ((not(member
                                 (concat 'm menu item)
                                 (eval(concat 'm menu))))
                           (msg  N  " You made an  error.  Try
again" N)
                           (change-menu-item))
                         (t (change-wording menu item)
                           (save-bckg-obs)
                           (msg  N N N "Here is the  new  menu
number " menu N N)
                           (printlit1  (nth  (- menu 2)  bckg-
obs-menu))
                           (msg  N N "REMINDER:  You must  use
TEACH DAVE MENU item 6 to make this change permanent" N N)
                           (change-menu-item]
```

This function is used to retrieve the wording for the menu item. It is used by both change-menu-item and add-menu-item. It uses readln which is a function located in "menu" which reads an entire line in as a string.

```
(defun change-wording (menu item)
 (let ((m-menu-item (concat 'm menu item)))
   (set m-menu-item nil)
   (msg N N "            ENTER NEW MENU ITEM WORDING")
   (msg  N  "Each  line should not exceed  3/4  of  the  screen
width")
   (msg  N  "Enter as many lines as required then  follow  them
with a + on a ")
   (msg N "line by itself." N N)
   (do ((line (readln) (readln)))
      ((equal line '| +|)(set m-menu-item (cons menu (cons item
                                                  (reverse  (eval
m-menu-item)))))))
```

```
      (set m-menu-item (cons line (eval m-menu-item)))
      (cond  ((eq  line '| help|) (exec "more  help21")(change-
wording menu item)))]
```

This function is very similar to change-menu-item except that is calls get-next-item to find the first unused item number on a particular menu.

```
(defun add-menu-item ()
   (msg  N  "******** Add observation & background  menu  items
**************" N)
   (msg N "The menus are numbered 2 through 11. Which one would
you like to add to?")
   (msg N "     Enter + when you are complete" N)
   (let ((menu (read)))
      (cond ((eq menu '+))
            ((eq  menu  'help)(exec  "more   help21")(add-menu-
item))
            ((not (member (concat 'm menu) m-nums))
                  (msg  N  "  You  made  an  error.  Try
again." N)
                  (add-menu-item))
            (t (msg N "     Here is the menu number " menu N N)
               (printlit1 (nth (- menu 2) bckg-obs-menu))
               (let ((item (get-next-item menu)))
                  (msg  N "The next menu item is number "  item
N)
                  (set (concat 'm menu)
                       (reverse(cons
                                  (concat 'm menu item)
                                  (reverse(eval(concat      'm
menu)))))))
                  (change-wording menu item)
                  (save-bckg-obs)
                  (msg  N  N N "Here is the new menu  number  "
menu N N)
                  (printlit1 (nth (- menu 2) bckg-obs-menu))
                  (add-menu-item]
```

This function is used by add-menu-item to find the first unused menu item number on a particular menu.

```
(defun get-next-item (menu)
   (do ((num 1 (add1 num))
        (m-menu (eval (concat 'm menu)) (cdr m-menu)))
       ((null m-menu) num) ]
```

The next two functions convert the lists of lists of identifiers described in m-nums above into the new value of bckg-obs-menu.

```
(defun save-bckg-obs ()
   (setq bckg-obs-menu (mapcar 'eval-lists (mapcar 'eval m-
nums]

(defun eval-lists (item)
   (do ((old-list (reverse item)(cdr old-list))
        (new-list nil (cons (eval (car old-list)) new-list)))
       ((null old-list)new-list]
```

APPENDIX L, MAKE FILE CHANGES PERMANENT

This file, called "permfile, contains functions which record the current value of several of the major lists in a file which will replace the former permanent value of those lists. The lists are the rules, clinical entity list, and major menus which govern the evaluation of Dave. The other entries in the Teach Dave Menu only change their lists in memory. Each of the lists in this section is located in a file by itself which is written over by these functions.

This function presents a menu which allows the user to select which one of the files is to be written.

```
(defun write-permfile ()
   (msg   N  N  "**************  MAKE  FILE  CHANGES  PERMANENT
**********************" N N)
   (msg "1. Save clinical entities" N)
   (msg "2. Save sign rules" N)
   (msg "3. Save observation and background menus" N)
   (msg "4. Save superqualifier/disqualifier rules" N)
   (msg "5. Save local score names" N)
   (msg "6. All of the above " N)
   (msg "7. Exit" N)
   (msg "      Enter number" N)
   (caseq (read)
      (1              (write-clinical-entities)(write-permfile))
      (2 (write-sign-rules)(write-permfile))
      (3 (write-obmenu)(write-permfile))
      (4 (write-sd-qualifiers)(write-permfile))
      (5 (write-local-score-list)(write-permfile))
      (6 (write-clinical-entities)
         (write-sign-rules)
         (write-obmenu)
         (write-sd-qualifiers)
         (write-local-score-list))
      (7 )
      (help (exec "more26")
         (write-permfile))
      (t (msg N "ERROR!!!!!! Enter 1 thru 6. Try again." N)
         (write-permfile)
```

This function writes the local-score list.

```
(defun write-local-score-list ()
   (setq my-output-port (outfile 'ls))
   (pp-form `(setq local-score-list (quote ,local-score-list))
my-output-port)
   (close my-output-port]
```

This function writes the clinical-entities.

```
(defun write-clinical-entities ()
   (setq my-output-port (outfile 'ce))
   (pp-form `(setq clinical-entities (quote ,clinical-
entities)) my-output-port)
   (close my-output-port]
```

This function writes the new observation & background menus.

```
(defun write-obmenu ()
   (setq my-output-port (outfile 'bomenu))
   (pp-form `(setq bckg-obs-menu (quote ,bckg-obs-menu)) my-
output-port)
   (close my-output-port]
```

This function writes the sign-rules.

```
(defun write-sign-rules ()
   (setq my-output-port (outfile 'rules))
   (pp-form `(setq sign-rules (quote ,sign-rules)) my-output-
port)
   (close my-output-port]
```

This function writes the sd-qualifier rules.

```
(defun write-sd-qualifiers ()
   (setq my-output-port (outfile 'sd-qualifiers))
   (pp-form `(setq sd-qualifiers (quote ,sd-qualifiers)) my-
output-port)
   (close my-output-port]
```

APPENDIX M, CHANGE CLINICAL ENTITIES AND LOCAL SCORES

This file, called "chce", contains the functions which change the clinical entities and local score lists.

This function presents a menu which allows the terminal user to choose to add to or change the clinical entity list.

```
(defun change-clinical-entities ()
   (msg    N   "***************   CHANGE   CLINICAL    ENTITIES
*********************" N N)
   (msg "These are the current clinical entities" N N)
   (mapcar 'printlit2 clinical-entities)
   (msg N "   Would you like to add or delete one of these?" N)
   (msg "     Enter      a = add" N)
   (msg "                d = delete " N)
   (msg "                + = neither" N)
   (caseq (read)
      ((a add) (add-clinical-entities))
      ((d delete) (remove-clinical-entities))
      ( + nil)
      (help (exec "more help22")(change-clinical-entities))
      (t (msg N "ERROR!!!!! Enter a d or +.  Try again." N)
         (change-clinical-entities))]
```

This function asks the terminal user for the name of a new clinical entity and adds it to the list of clinical entities and prints the new list.

```
(defun add-clinical-entities ()
   (msg    N    "*****************   add   clinical    entities
*********************" N N)
   (msg  N  N N "Enter the new one using  - instead  of  spaces
between words." N)
   (let ((answer (read)))
     (cond ((eq answer 'help)(exec "more help22")
                            (add-clinical-entities))
           (t (setq clinical-entities
                    (sort(cons answer clinical-entities)))
              (msg N N N "These are the new clinical entities."
N N)
              (mapcar 'printlit2 clinical-entities]
```

This function asks the name of the clinical entity to be

removed and removes it from the clinical entity list. It then prints the new list.

```
(defun remove-clinical-entities ()
   (msg   N   "*****************   delete   clinical   entities
*******************" N N)
   (msg N N "Which entity should be removed?" N)
   (let ((answer (read)))
     (cond ((eq answer 'help)(exec "more help22")
                            (remove-clinical-entities))
           (t (setq clinical-entities
                          (remove answer clinical-entities))
             (msg N N N "These are the new clinical entities."
N N)
             (mapcar 'printlit2 clinical-entities]
```

This function presents a menu which allows the user to coose to *add to or remove* identifiers from the local score list.

```
(defun change-local-score-list ()
   (msg    "***************    CHANGE    LOCAL    SCORE    LIST
*******************" N N)
   (msg "These are the current local scores" N N)
   (mapcar 'list-scores local-score-list)
   (msg  N N "   Would you like to add or delete one of these?"
N)
   (msg "     Enter      a = add" N)
   (msg "                d = delete " N)
   (msg "                + = neither" N)
   (caseq (read)
      ((a add) (add-local-scores))
      ((d delete) (remove-local-scores))
      ( + nil)
      (help (exec "more help23")(change-local-score-list))
      (t (msg N "ERROR!!!!! Enter a d or +.  Try again." N)
         (change-local-score-list)]
```

This function asks for a description of the new score subtest. It will accept a one line phrase using readln which is contained in "menu".

```
(defun add-local-scores ()
   (msg      "*****************      add      local      scores
**********************" N N)
   (msg N N N "Enter the description of the new score." N)
   (msg  "It  should  be a word or   phrase  contained  on  one
line.")
   (let ((answer (readln)))
      (cond ((eq answer '| help|)(exec "more help23")
                                    (add-local-scores))
            (t (get-variable answer]
```

This function receives description of the new rule from the function above and asks the terminal user for the abbreviation of the new score. If the abbreviated test score name has not been used elsewhere, then it adds the new identifier and its description to the local scores list.

```
(defun get-variable (description)
        (msg "Now enter a one word abbreviation for the same
score" N)
        (msg "It may be the same word or an abbreviation." N)
        (let ((variable (read)))
           (cond ((or (member variable (mapcar 'car  all-score-
list))
                      (member  variable (mapcar 'car  all-prev-
score-list))
                      (member   variable  (mapcar  'car  local-
score-list)))
                           (msg N "ERROR!!!  This word is being
used. " N)
                           (msg "      Try again." N)
                           (get-variable description))
                 ((eq variable 'help)(exec "more help23")
                           (get-variable description))
                 (t (setq local-score-list
                          (cons (list variable description)
                                local-score-list)))))
   (msg N N N "These are the new local scores." N N)
   (mapcar 'list-scores local-score-list]
```

The next two functions are used to remove an identifier and its description from the local score list. This function calls

remove-score-var to remove the entry which it has gotten from the terminal user. It displays the new local score list.

```
(defun remove-local-scores ()
   (msg    N    "*****************    delete    local    scores
*******************" N N)
   (msg N N "Which score name should be removed?" N)
   (let ((answer (read)))
        (cond ((eq answer 'help)(exec "more hlep23")
                                (remove-local-scores))
              (t (remove-score-var answer)
                (msg N N N "These are the new local score names.
" N N)
                (mapcar 'list-scores local-score-list)]
```

This function searches through the local score list to find a match with v, the variable name, and the car of the local score lists. If a match is found, it removes the entire entry from the local score list.

```
(defun remove-score-var (v)
   (do ((score local-score-list (cdr score)))
       ((null score))
       (cond ((eq v (caar score))
                  (setq local-score-list (remove (car score)
local-score-list]
```

APPENDIX N, PRODUCE OUTPUT REPORT

This file, called "out", contains the functions which produce the report in a readable form. For more detail, see Appendix T, Inference Engine Algorithm. The output lists which are associated with each of the fired clinical entities as identifiers have already been produced. The three type scores which are a concatenation of the clinical entity name and the type of evidence have already been calculated. Also contained are the modified functions which produce a totally fired report as in Item 8 of the Evaluation Menu.

This function prints a heading, calls three functions to sort and print the bulk of the report and prints the scores and observations which were used in the evaluation.

```
(defun produce-report ()
   (let ((old-port poport))
      (setq poport (outfile 'report))
      (msg                                                    N
"*****************************************************************'
N)
      (msg  N  N  "            DAVE'S  EVALUATION  OF  POSSIBLE
CLINICAL ENTITIES" )
      (msg                    N                   N                   N
"*****************************************************************'
      (sort-fired-ce)
      (mapcar     'output-clinical-entities     fired-clinical-
entities)
      (other-observations)
      (msg N N N N "* * * * * * * * * * * * * WAIS SCORES * * * *
* * * * * * * * * *" N N)
      (mapcar 'printlit4 score-list)
      (cond ((neq prev-i 0)
                  (msg  N N N "* * * * * * * * * * * * *  PREVIOUS
WAIS SCORES * * * * * * * " N)
                  (mapcar 'printlit4 prev-score-list)))
      (cond ((null local-score-list))
            (t (msg N N N "* * * * * * * * * * * * * LOCAL SCORES
* * * * * * * * * * *" N)
               (mapcar 'printlit4 local-score-list)))
      (msg N N N "* * * * * * * * * OBSERVATIONS AND BACKGROUND
INFORMATION * * * *" N)
```

```
      (printlit1a obs-list)
      (close poport)
      (setq poport old-port]
```

This function is used to regain control of the screen if there is an error during the production of the report. At this time it is only used by the programmer during debugging.

```
(defun come-back ()
      (close poport)
      (setq poport old-port]
```

The next two functions are used to put the clinical entities into order according to the evidence scores. By multipling the observation score by 100 and the WAIS score comparison by 10, the types of evidence are considered in order of their importance.

```
(defun sort-fired-ce ()
   (let  ((fce-score-list (mapcar  'make-fce-score-list  fired-
clinical-entities)))
      (sortcar fce-score-list '))
      (setq  fired-clinical-entities  (mapcar 'cadr  fce-score-
list)]


(defun make-fce-score-list (ce)
   (list (+ (* 100 (eval (implode (append (explode ce) (explode
'observe)))))
            (*  10 (eval (implode (append (explode ce) (explode
'score)))))
            (eval   (implode  (append  (explode  ce)   (explode
'bckgrnd)))))
         ce]
```

This function is used to print the output for a clinical entity associated with "ce". Each of the evidence scores are printed. Output-clause is called to print the description of the supported rules and again for the might-add rules.

```
(defun output-clinical-entities (ce)
   (cond ((equal (eval ce) nil))
         (t
          (blank-line 10)
          (msg     "****************************      "    ce    "
*************************")
          (msg   N   N  "The  evidence   score   for   examiner's
observations was ")
          (print  (eval  (implode  (append  (explode  ce)(explode
'observe)))))
          (msg N "The evidence score for WAIS test scores was ")
          (print  (eval  (implode  (append  (explode  ce)(explode
'score)))))
          (msg  N "The evidence score for background  information
was ")
          (print  (eval  (implode  (append  (explode  ce)(explode
'bckgrnd)))))
          (msg N N N N "            The supported rules for " ce "
are as follows" N N N)
          (mapcar 'output-clause (reverse (eval ce)))
          (cond ((equal nil (eval (implode (append
                                              (explode ce)
                                              (explode     'might-
add))))))
                (t (msg N N N N N "            The might add rules
for " ce
                        " are as follows" N N N)
                   (mapcar 'output-clause (eval (implode
                                          (append(explode
ce)(explode 'might-add]
```

This is the function used to print the other-observations title and call output-clause to print the rules which indicate entities which are not on the fired clinical entities list.

```
(defun other-observations ()
   (blank-line 10)
   (msg   "****************************     other  observations
***************************" N N N)
   (mapcar 'output-clause other-fired-rules]
```

This is the function which prints the bulk of the output for the final report. It receives in 1 the list of all sign-rule evidence descriptions, hypotheses, and superqualification

evidence descriptions. It continues to call itself with its own cdr until the output is complete.

```
(defun output-clause (l)
   (cond ((null l))
         (t (caseq (caar l)
                (superqualified (princ 'superqualified)
                                (terpr)
                                (mapcar 'printlit2a (cdar l))
                                (blank-line 3))
                (score          (msg "     score")
                                (printlit3 (cdar l))
                                (output-clause (cdr l)))
                (not            (msg "    the following item  is
not present" N)
                                (output-clause (cdar l)))
                (observe        (msg  "     observation")
                                (find-observe
                                    (nth     (- (cadar  l)   2)
bckg-obs-menu)
                                    (caddar l))
                                (terpr)
                                (output-clause (cdr l)))
                (bckgrnd        (msg "     background")
                                (find-observe
                                    (nth      (- (cadar    1)2)
bckg-obs-menu)
                                    (caddar l))
                                (terpr)
                                (output-clause (cdr l)))
                (main           (msg N "Main hypothesis   ")
                                (cond ((neq '| none| (cadadar l))
                                  (msg (cadadar l) "  ")))
                                (msg (caddar l) N)
                                (output-clause (cdr l)))
                (secondary      (cond ((neq 'none (cadar l))
                                  (msg      "          secondary
hypothesis   ")
                                  (cond  ((neq  '| none|  (cadadar
l))
                                                (msg (cadadar  l)
"  ")))
                                  (msg (caddar l) N)))
                                (output-clause (cdr l)))
                (additional     (cond ((neq '| none| (cadar l))
                                (msg  "     additional support?
" (cadar l) N)))
                                (output-clause (cdr l))]
```

This function is used to locate an item number on a menu and print its English language description on the output report.

```
(defun find-observe (menu-list item)
   (cond ((equal (cadar menu-list) item)
                 (princ (caddar menu-list))
                 (terpr))
         (t (find-observe (cdr menu-list) item]
```

This function in a modification of produce-report which causes a report to be produce with every rule printed as if the evidence was present.

```
(defun produce-totally-fired-report ()
   (let ((old-port poport))
        (setq poport (outfile 'report))
        (msg                                                    N
"*************************************************************"
N)
        (msg  N  N  "                    DAVE'S  REPORT  OF  ALL
RULES" N)
        (msg  "       This  is not an actual  evaluation  but  a
report which makes all rules" N)
        (msg    "          true.    It    does    not    consider
superqualification or disqualification" N)
        (msg                              N                     N
"*************************************************************"
N)
        (sort-fired-ce)
        (mapcar    'output-clinical-entities     fired-clinical-
entities)
        (other-observations)
        (close poport)
        (setq poport old-port)]
```

APPENDIX O, CHANGE SIGN-RULE

This file, called "chrule", contains functions which are used to add or change the sign-rules. The functions for adding evidence are also used by the "chsdq" file for changing the sd-qualifier rules. See Appendix R for the algorithm to change a sign-rule.

This function presents a menu which allows the user to choose between viewing the rules currently in memory or the permanent ones on file, or adding or changing a rule. If one is to be changed the routine asks which rule is to be scanned first.

```
(defun change-rule ()
   (msg    N    N    N    "*****************************    RULES
************************" N N)
   (msg "1. View latest permanent rules on screen" N)
   (msg "2. View current rules on screen" N)
   (msg "3. Search rules for possible changes " N)
   (msg "4. Add new rule" N)
   (msg "5. Exit" N)
   (msg "      Enter menu item number.")
   (caseq (read)
       (1 (exec "more rules"))
       (2 (pp-form sign-rules))
       (3  (msg N "With which rule number will you begin scan?"
N)
           (let ((rule-num (read)))
              (cond ((eq rule-num 'help)
                         (exec "more help24j")(change-rule))
                    ((numberp  rule-num)
                         (get-inside-list rule-num))
                    (t   (msg N "Rule number must be a number"
                          N "Try again" N)
                         change-rule)))))
       (4 (add-rule))
       (5 nil)
       (help (exec "more help24a")(change-rule))
       (t (msg N "ERROR!!!! Enter 1 thru 5. Try again." N N N)
          (change-rule))]
```

This function is used to tear appart the sign-rule list to find the target rule which is to be changed. It uses check-target to get the new rule if the target rule is the one to be changed. It then puts the list back toghther in the correct order.

```
(defun get-inside-list (r)
   (do ((front nil (cons target front))
        (target (car sign-rules) (car end))
        (end (cdr sign-rules) (cdr end))
        (rulenum 1 (add1 rulenum)))
      ((or (null target)
           (and ()= rulenum r)
                (setq new-rule (check-target target))))
                   (cond ((eq new-rule '+)
                           (msg  N " Abandoned substitution"  N
))
                         (t (setq sign-rules (append  (reverse
front)
                                                     new-rule
                                                     end))]
```

This function prints the target rule to see if it is the one to be changed. At this point 1 and target are the same. If it is the correct rule, it calls check-susstitute-rule to get the new rule.

```
(defun check-target (1)
   (msg "Is this the one to change?" N (pp-form target) N )
   (msg "        Enter y to change this rule" N)
   (msg "              n to get next rule" N)
   (msg "              + to quit search  " N)
   (caseq (read)
       (+ '+)
       (help (exec "more help24b")(check-target 1))
       (y (check-substitute-rule]
```

The following two functions are used together. They call get-substitute-rule to get the new rule and then print it for the user to decide if it is what he wanted. If the rule is

incorrect the second function will call the first to begin again. They are separated for error and help facility recovery.

```
(defun check-substitute-rule ()
   (let ((new-rule (get-substitute-rule (cadr target))))
        (check-substitute-rule-2)]

(defun check-substitute-rule-2 ()
       (msg N "Is this new rule correct?"N (pp-form new-rule) N
)
       (msg "       Enter y to make substitution" N)
       (msg "             n to redo rule" N)
       (msg "             + to abandon the substitution. " N)
          (caseq (read)
             (y new-rule)
             (+ '+)
             (help (exec "more help24c")
                   (check-substitute-rule-2))
             (t (check-substitute-rule)]
```

This function calls two other functions to get the evidence and the hypothesis. If they are both non-nil then it puts them together with the appropriate key words to form the new rule.

```
(defun get-substitute-rule (rule-number)
   (blank-line 5)
   (msg "                    GET SUBSTITUTE RULE" N N)
   (msg  "Select one or more checks from the list  below.  They
should be in " N)
   (msg  "order  of importance.   All of the evidence  must  be
present in order to be" N)
   (msg "listed in the evaluation report.  Designation of  this
rule for the " N)
   (msg "might-add list may be done next followed by listing of
hypotheses. " N N)
   (let ((evidence (get-rule-evidence))
         (hypo (get-hypotheses)))
               (cond ((and evidence hypo)
                           (list (cons 'rule
                                       (cons rule-number
                                             (append evidence
                                                     hypo]
```

The following two functions are used together. They call two other functions to get the hypothesis and the evidence and a third to get the next unused rule number. They put the parts of the rule together and print it for the user to determine if it is correct. If it is, then the new rule is added to the end of the sign-rules and the user is given the option of adding another rule or exiting.

```
(defun add-rule ()
   (blank-line 10)
   (msg "                          ADD RULE" N N)
   (msg "Select one or more evidence checks from the list
below. They should be in " N)
   (msg "order of importance. All of the evidence checks must
be present in order to be" N)
   (msg "listed in the evaluation report. Designating rules for
the " N)
   (msg "might-add list may be done next followed by
hypotheses." N N)
   (let ((evidence (get-rule-evidence))
         (hypo (get-hypotheses)))
         (cond ((and evidence hypo)
        (add-rule-2]

(defun add-rule-2 ()
                  (let ((new-rule (cons 'rule
                                        (cons   (add1  (cadar  (last
sign-rules)))
                                              (append evidence
                                                      hypo)))))
                     (msg N N "Is this new rule correct?" N)
                     (pp-form new-rule)
                     (msg N N "       Enter y if correct." N)
                     (msg "             n to redo rule." N)
                     (msg "             + to abandon this rule."
N)
                     (caseq (read)
                        (y (setq sign-rules (append sign-rules
                                                    (list new-rule)))
                        (n (add-rule))
                        (help   (exec  "more  help24c")
                                (add-rule-2))
                        (+ '+))]
```

This function prints the menu contained in evidence-options below. It will build a list of evidence checks by calling other functions and cons-ing them together until item 6 or 7 is used. The value of the evidence rule is returned in the value of the function itself.

```
(defun get-rule-evidence ()
  (msg N "        ADD EVIDENCE TO RULE" N)
      (mapcar 'printlit2 evidence-options)
      (msg "Select the rule evidence to be used ." N)
      (caseq (read)
         (1 (cons (get-score-rule)(get-rule-evidence)))
         (2 (cons (get-observe-evidence)(get-rule-evidence)))
         (3  (cons (list 'not (get-observe-evidence))(get-rule-
evidence)))
         (4 (cons (get-bckgrnd-evidence)(get-rule-evidence)))
         (5 (cons (list 'not  (get-bckgrnd-evidence))(get-rule-
evidence)))
         (6 (get-certain))
         (7 nil)
         (help (exec "more help24d")(get-rule-evidence))
         (t (msg N "ERROR!!!!! Enter 1 thru 7. Try again." N)
            (get-rule-evidence]
```

This is the menu used by the get-rule-evidence above.

```
(setq evidence-options
        '("1. WAIS score comparison."
          "2. Observations"
          "3. Absence of an observation"
          "4. Background information."
          "5. Absence of a piece of background information"
          "6. Identify rule for might add list."
          "       (must be last evidence and must follow one or
more of the three above"
          "7. End of evidence checks for this rule.")]
```

This function allows input of a menu and item number for addition of observation evidence. It allows the user to view the appropriate menu if needed and checks to see that the entry is valid. The new evidence is returned in the value of the function. eg. (observe 3 5).

```
(defun get-observe-evidence ()
   (msg N "                    OBSERVATION EVIDENCE" N N)
   (msg "Which menu number do you want in your rule?" N)
   (msg  "      Enter + to view menus." N)
      (let ((menu (read)))
         (cond    ((eq    menu     '+)(show-obs-menu)(get-observe-
evidence))
                  ((eq menu 'help)(exec "more help24e")
                                  (get-observe-evidence))
               (t
                 (msg  N "Which item on menu number " menu "  do
you want?" N)
                 (let ((item (read)))
                    (cond  ((nth (- item 1)(nth (- menu 2) bckg-
obs-menu))
                                 (msg N " This is your selection:  "
N)
                                 (printlit1b (nth
                                              (- item 1)
                                              (nth (- menu 2)  bckg-
obs-menu)))
                                 (msg "       Is it correct?  (Enter
y/n)")
                                 (caseq (read)
                                   (y (list 'observe menu item))
                                   (help (exec "more help24e")
                                         (get-observe-evidence))
                                   (t (get-observe-evidence))))
                           ((eq item 'help)(exec "more help24e)
                                   (get-observe-evidence))
                           (t  (msg  N "This item is not  on  the
menu." N)
                                 (get-observe-evidence)]
```

This function works exactly the same as get-observe-evidence except the key word is different. eg. (bckgrnd 8 10).

```
(defun get-bckgrnd-evidence ()
   (msg N "                BACKGROUND EVIDENCE" N N )
   (msg "Which menu number do you want in your rule?" N)
   (msg "      Enter + to view menus." N)
      (let ((menu (read)))
         (cond     ((eq    menu    '+)(show-obs-menu)(get-bckgrnd-
evidence))
                   ((eq menu 'help)(exec "more help24e")
                                   (get-bckgrnd-evidence))
                (t
                 (msg  N "Which item on menu number " menu "  do
you want?" N)
                 (let ((item (read)))
```

```
                    (cond ((nth (- item 1)(nth (- menu 2)  bckg-
obs-menu))
                           (msg N " This is your selection:  "
N)
                           (printlit1b (nth
                                       (- item 1)
                                       (nth  (- menu 2) bckg-
obs-menu)))
                           (msg "       Is it correct?  (Enter
y/n)")
                           (caseq (read)
                             (y (list 'bckgrnd menu item))
                             (help (exec "more help24e")
                                   (get-bckgrnd-evidence))
                             (t (get-bckgrnd-evidence))))
                          ((eq item 'help)(exec "more help24e")
                                   (get-bckgrnd-evidence))
                          (t  (msg  N "This item is not  on  the
menu." N)
                           (get-bckgrnd-evidence)]
```

This function is used by the two functions above to display a menu to the user for reference.

```
(defun show-obs-menu ()
   (msg N "Which menu would you like to see?" N)
   (let ((menu (read)))
       (cond ((member menu '(2 3 4 5 6 7 8 9 10 11))
                   (printlit1 (nth (- menu 2) bckg-obs-menu)))
             ((eq menu 'help)(exec "more help24e")
                   (show-obs-menu))
             (t (msg N "This item is not on the menu." N)
                (show-obs-menu]
```

The following two functions are used by various functions to format output.

```
(defun printlit1b (l)
    (msg N (car l) B B (cadr l) B (printlit2b (cddr l)) N N N]

(defun printlit2b (l)
   (cond ((null l))
         (t (msg (car l) N)
            (printlit2b (cdr l]
```

This function is used to get the part of the evidence clause which will mark the rule for the might-add list. The clinical entity listed after the key word "certain" must be contained on the clinical-entities list. The value of the evidence is returned in the function. eg. (certain psychosis).

```
(defun get-certain ()
   (msg  N  N "                    DESIGNATE RULE  FOR  MIGHT  ADD
LISTS" N)
   (msg N "Now enter one of the primary clinical entities under
study. " N)
   (msg  "This rule will be added to its might add list if  the
rule's " N)
   (msg  "evidence  checks are true and the clinical entity  if
found to be present." N)
   (let ((ce (read)))
      (cond ((member ce clinical-entities) (list (list 'certain
ce)))
            ((eq ce 'help)(exec "more help24d")(get-certain))
            (t  (msg N "ERROR!!!!!!  " ce " is not one  of  the
clinical entities.")
                (msg N "Here are the current clinical entities."
N)
                (mapcar 'printlit2 clinical-entities)
                (msg  N N "Do you want to abandon this piece  of
evidence? (enter y/n)" )
                (caseq (read)
                       (y nil)
                       (help (exec "more help24d")(get-certain))
                       (t (get-certain]
```

This function calls two others to get the two parts of the WAIS score evidence. It puts those together with the key word "score" and returns the evidence value in the function.

```
(defun get-score-rule ()
   (cons     'score     (list     (get-score-evidence)(get-score-
description)]
```

This function gets the WAIS score evidence. It first has to determine the level of complexity of each sub-expression in

the piece of evidence. It prints a menu which allows the user to enter the appropriate number of left parentheses. In this case parentheses are only used to surround the data items involved with a binary AND/OR clause. See Appendix R for further explanation. This function will be used to obtain both sides of and expression of a AND/OR clause. It will continue to appear as long as the user enters Item 2.

```
(defun get-score-evidence ()
   (msg N N "         COMPLEXITY OF WAIS SCORE COMPARISONS")
   (msg N "1. WAIS score comparison.")
   (msg N "2. (  Left parenthesis" )
   (msg N "        Only use to surround AND/OR clause" N)
   (msg N "     eg.(a ) 50   AND ")
   (msg N "                  (vmean )= pmean   OR   i ) ds + 2))"
)
   (msg N "Enter menu item number." B)
   (caseq (read)
      (1 (get-relative-operator))
      (2 (get-logical-operator))
      (help (exec "more help24f")(get-score-evidence))
      (t (msg N "ERROR!!!! Enter 1 or 2. Try again." N)
         (get-score-evidence)]
```

The following two functions get a value of a logical expression such as (X and Y) where X and Y can be simple relational expressions or other logical expressions. They are split into two only for the purpose of error recovery. The value of the expression is returned in the function.

```
(defun get-logical-operator ()
   (let ((first (get-score-evidence)))
      (get-log-op-menu]

(defun get-log-op-menu ()
      (msg N N "                    LOGICAL OPERATORS" N N)
      (msg "1. AND" N)
      (msg "2. OR" N)
      (msg "    Enter menu item number" B)
      (caseq (read)
         ((1 AND and) (list 'and (get-score-evidence) first))
         ((2 OR or) (list 'or (get-score-evidence) first))
```

```
      (help (exec "more help24g")(get-log-op-menu))
      (t (msg N "ERROR!!!!!  Enter 1 2 or 3. Try again." N)
         (get-log-op-menu]
```

The following two functions obtain a relational expression of the form R) S where R and S are simple data items or arithmetic expressions as described below. The two functions are divided only for the purpose of error recovery. The value of the expression is returned in the function.

```
(defun get-relative-operator ()
   (let ((first (get-arithmetic-operator)))
       (get-rel-op-menu)]


(defun get-rel-op-menu ()
   (msg N N "                RELATIVE OPERATORS FOR  COMPARISON"
N N)
   (msg  "You  will  now  use one  of  the  following  relative
operators to " N)
   (msg  "compare  the item or arithmetic expression  that  you
just " N)
   (msg "entered with the next one." N N)
   (msg "1.      =       Equal" N)
   (msg "2.      (        Less than" N)
   (msg "3.      )        Greater than" N)
   (msg "4.      (=       Less than or equal" N)
   (msg "5.      )=       Greater than or equal" N)
   (msg "6.      ()       Not equal" N)
   (msg "     Enter the appropriate number." N)
   (caseq (read)
      ((1 =) (list 'eq first (get-arithmetic-operator)))
      ((2 () (list 'lessp first (get-arithmetic-operator)))
      ((3 )) (list 'greaterp first (get-arithmetic-operator)))
      ((4 (=) (list '(= first (get-arithmetic-operator)))
      ((5 )=) (list ')= first (get-arithmetic-operator)))
      ((6 ()) (list 'neq first (get-arithmetic-operator)))
      (help (exec "more help24g")(get-rel-op-menu))
      (t (msg N "ERROR!!!  Enter 1 thru 6. Try again")
         (get-rel-op-menu)]
```

The following two functions obtain a simple data item as described below or an arithmetic expression of the form A + B where A and B are simple data items or other arithmetic

expressions. The two functions are divided for error recovery. The value of the expression is returned in the function.

```
(defun get-arithmetic-operator ()
   (let ((first (get-item)))
      (get-ar-op-menu]

(defun get-ar-op-menu ()
  (msg N N N N "              ARITHMETIC OPERATORS (optional)" N
N)
  (msg  "Now I must know if you want to do any arithmetic  with
the item" N)
  (msg "which you just entered and another item on this side of
the " N)
  (msg "comparison operator .")
  (msg  N  "Arithmetic operations will be executed in order  of
entry. " N)
  (msg N "1.     +      Addition" N)
  (msg "2.     -      Subtraction" N)
  (msg "3.     *      Multiplication" N)
  (msg "4.     /      Division" N)
  (msg "5.     none" N)
  (msg N "             Enter the appropriate number." N)
  (caseq (read)
     ((1 +) (list 'add first (get-arithmetic-operator)))
     ((2 -) (list 'difference first (get-arithmetic-operator)))
     ((3 *) (list 'times first (get-arithmetic-operator)))
     ((4 /) (list 'quotient first (get-arithmetic-operator)))
     ((5 n no none) first)
     (help (exec "more help24g")(get-ar-op-menu))
     (t (msg N "ERROR!!!!! Enter 1 thru 5.  Try again." N)
        (get-ar-op-menu]
```

This function obtains a value of a data item to be used in the evidence. It is either a number or one of the elements on one of the three list of score identifiers. It will display the valid score names if requested or if an invalid one is entered.

```
(defun get-item ()
  (msg N "            DATA ITEM" N)
  (msg  "Enter  a number or abbreviated WAIS score name  to  be
used." N)
  (msg "(Enter + for current WAIS scores,  ++ for previous; +++
for local scores.)" N)
  (let ((item (read)))
     (cond   ((eq  item  '+)(mapcar  'list-scores   all-score-
```

```
list)(get-item))
            ((eq item '++)(mapcar 'list-scores  all-prev-score-
list)(get-item))
            ((eq  item  '+++)(mapcar 'list-scores  local-score-
list)(get-item))
            (help (exec "more help24g")(get-item))
            ((numberp item) item)
            ((find-variable item all-score-list) item)
            ((find-variable item all-prev-score-list) item)
            (t (msg N "ERROR!!!! Try again." N)(get-item]
```

This function is used by get-item to check to see if the data item entered is on one of the three lists.

```
(defun find-variable (item l)
   (cond ((null l) nil)
         ((eq item (caar l)))
         (t (find-variable item (cdr l)]
```

This function is used to format the list of scores and their descriptions on the screen.

```
(defun list-scores (l)
     (msg N (car l) " = " (cadr l))]
```

The following two functions are used to get a list of string values which describe the evidence which has just be entered. They use the readln function located in "menu". It will continue to ask for more lines until a "+" is entered on a line by itself. Spaces are padded to the begining of each string so that it will appear on the output report properly.

```
(defun get-score-description ()
   (msg N N N N "              EVIDENCE DESCRIPTION" N N)
   (msg  "Now you need to give an English language  description
of the evidence" N)
   (msg  "which you have just entered.  This description  will
apear on the" N)
   (msg "report." N)
   (msg "You may use as many lines as needed.  Each line should
not exceed " N)
```

```
   (msg  "2/3  of  the width of  the  screen.  This  should  be
followed by a + on a " N)
   (msg "line by itself." N N)
   (msg "For example:" N)
   (msg "digit span or arithmetic score ( verbal mean" N)
   (msg "or picture completion ( verbal mean." N)
   (msg "+" N N)
   (msg "Now enter new description" N)
   (let ((first (readln)))
    (cond ((eq first '! help!)(exec "more hlep24h")
                              (get-score-description))
          (t (let ((line (cons first (get-next-line))))
             (msg N N N N "Is this what you wanted? " N)
             (msg "            ")
             (mapcar 'printlit2 line)
             (msg "Enter y/n. ")
             (caseq (read)
               (y line)
               (help (exec "more help24h")
                     (msg N "re-enter description" N N)
                     (get-score-description))
               (t (msg N N "Try again" N)
                     (get-score-description]


(defun get-next-line ()
   (let ((line (readln)))
       (cond ((eq line '! +!) nil)
             ((eq line '! help!)(exec "more help24h")
                                (msg N "Enter next line or +" N)
                                (get-next-line))
             (t  (cons  (concat "             "  line)
                        (get-next-line)]
```

The remaining functions obtain the value of a list of hypotheses.

This function first offers the user a chance to abandon the new rule. If not, it will call get-hypothesis-2 to obtain a value.

```
(defun get-hypotheses ()
   (msg  N N N N "Do you need to abandon this new rule at  this
time? " N)
   (msg "Enter y/n     " N)
   (caseq  (read)
    (y nil)
    (help (exec "more help24i")(get-hypotheses))
    (t (msg N N N N "                              HYPOTHESES" N)
     (msg  N  "Now you will enter one or more hypotheses  which
```

```
may be" N)
     (msg  "present when these evidence checks are found to  be
true.  They should" N)
     (msg "be entered in order of importance.   Only the  first
one which " N)
     (msg  "matches  a major clinical hypothesis will count  in
determining" N)
     (msg "the most probable clinical entity." N N N)
     (get-hypotheses-2)]
```

This function first asks if this hypothesis will be the last one. It calls three functions to get the three parts of the hypothesis and puts them together with their key words to form a hypothesis clause. If there are to be more hypotheses then it calls itself again.

```
(defun get-hypotheses-2 ()
   (msg N N "                    BEGIN NEW HYPOTHESIS" N)
   (msg  N  "Will this be the final hypothesis for  this  rule?
Enter y/n." )
   (let ((answer (read)))
     (cond ((eq answer 'y)
                 (list (cons 'hypothesis
                             (list (get-main)
                                   (get-secondary)
                                   (get-additional)))))
           ((eq answer 'help)(exec "more help24i")
                             (get-hypotheses-2))
           (t (cons (cons 'hypothesis
                          (list (get-main)
                                (get-secondary)
                                (get-additional)))
                    (get-hypotheses-2)]
```

This function gets the first part of the hypothesis. It puts the key word, the value of the function get-qualifier and the entity entered by the user together in a clause. eg. (main (qualifier probably) mental-retardation). It will print the list of clinical-entities if requested.

```
(defun get-main ()
   (msg N N N N "          MAIN CLINICAL ENTITY OF  HYPOTHESIS"
N)
```

```
   (msg N "First identify the clinical entity. It may be one of
the main" N)
   (msg  "ones being evaluated or another such as depression or
" N)
   (msg "inadequate-personality.  It should be one word even if
" N)
   (msg  "you  need  a - to glue parts  together.  Do  not  use
quotation marks." N)
   (msg "Enter desired clinical entity." N)
   (let ((ce (read)))
     (cond ((eq ce 'help)(exec "more help24i")(get-main))
           (t (cons 'main (list (get-qualifier) ce))]
```

This function obtains the secondary part of the hypothesis. It puts the key word "secondary" together with the value of the function get-qualifier and the entity entered by the user together in a list. If the entity is "none" then the qualifier is not added. eg. (secondary (qualifier likely) schizophrenia) or (secondary none)

```
(defun get-secondary ()
   (msg  N  N  N  N "          SECONDARY  CLINICAL  ENTITY  OF
HYPOTHESIS" N)
   (msg N "The secondary clinical entity in this hypothesis  is
a more specific")
   (msg N "diagnosis of the main clinical entity. Schizophrenia
is a specific")
   (msg  N "type of psychosis for example.  It must also be one
word and you " )
   (msg N "will be able to further qualify it later.")
   (msg  N  "Enter the word none if there is  not  a  secondary
clinical-entity." N)
   (let ((ce (read)))
      (cond ((eq ce 'none) '(secondary none))
            ((eq ce 'help)(exec "more help24i")
                          (get-secondary))
            (t (cons 'secondary (list (get-qualifier) ce]
```

This function obtains a qualifier list. It consists of the key word "qualifier" and a word or phrase which is stored as a string. It uses the function readln located in "menu".

```
(defun get-qualifier ()
   (msg N N N "           QUALIFIER OF CLINICAL ENTITY" N)
   (msg  N  "The  qualifier is a word or phrase  which  further
describes the " )
   (msg N "clinical entity or gives a measure of certainty that
this ")
   (msg N "clinical entity is present if the evidence checks in
this rule are true." N)
   (msg N "probably endogenous    and")
   (msg N "sometimes conditions for")
   (msg N "are examples of qualifiers.   Enter the word none if
none exist.")
   (msg N "Enter a word or one phrase on one line." N)
   (let ((answer (readln)))
     (cond ((eq answer '| help|)(exec "more help24i")
                                  (get-qualifier))
           (t (list 'qualifier answer]
```

This function obtains the third part of the hypothesis, the additional list. It consists of the key word "additional" and a word or phrase which is stored as a string. It uses the function readln located in "menu".

```
(defun get-additional ()
   (msg N N N "             ADDITIONAL SUPPORTING EVIDENCE" N)
   (msg  N  "This is a word or one phrase which if  true  would
lend greater ")
   (msg N "evidence that this clinical entity is  present.   It
is not evaluated")
   (msg N "by Dave but is given to the examiner as further data
to examine.")
   (msg  N "Enter a word or phrase within quotation  marks.  If
none enter the ")
   (msg N "word none."N)
   (let ((evidence (readln)))
     (cond ((eq '| none| evidence)(list 'additional evidence))
           ((eq evidence '| help|)(exec "more help24i")
                                  (get-additional))
           (t
        (msg  N  N  N N "Is this what you  want  as  additional
supporting evidence?" N)
        (msg N evidence)
        (msg N "      Enter y/n ")
        (caseq (read)
              (y (list 'additional evidence))
              (help (exec "more help24i")(get-additional))
              (t (msg N N "    Try again." N N)
                 (get-additional)]
```

APPENDIX P, LIST OF ALL FILES

This is a listing of the files used by Dave and a list of appendices which contain the code contained in the file. There is a special file called "dave" which contains LISP and the first five files in the list below. The root names of the files are given below but in many cases the code is contained in a file with a ".l" suffix and the compiled code is contained in a file with a ".o" suffix.

FILE NAME	APPENDIX
menu	F
driver	G
sdq	H
obsmenu	J
out	N

The above executable version of the above files are contained in the file called "dave". They accept input of scores and menu responses and produce the report.

ce	E
ls	none
bomenu	C
sd-qualifiers	D
rules	B

This is a list of the rules, clinical entities, local scores (if any), and menus. They are loaded when Dave is started.

chsdq	I
chobs	K
permfile	L
chce	M
chrule	O

These are files used by the Teach Dave Menu to alter the second group of lists above. They are only loaded when their menu item is selected.

APPENDIX Q, THE USERS MANUAL

TABLE OF CONTENTS

SECTION PAGE

1. INTRODUCTION 140

2. GETTING STARTED 140

3. ERROR RECOVERY 142

4. ENTERING VALUES OF SUBTEST SCORES 143

5. OBSERVATION & BACKGROUND MENUS 144

6. THE REPORT 144

7. THE RULES 147

8. TEACH DAVE MENU 151

SECTION 1. INTRODUCTION

This users manual is designed to give very basic instructions concerning the operation of Dave. Much of the detailed knowledge required to use this system is either given to the operator automatically on the screen or can be obtained using the "help" facility. It does not attempt to explain psychological terms.

Dave is a rule based expert system which is designed to be an aid for the evaluation of the Wechsler Adult Intelligence Scales and related information. The related information is obtained by presenting you with a series of menus. These contain a list of statements which may or may not be true of your observations during the administration of the test and of his known background information. You are probably well aware of the process of manually evaluating this information. Dave produces a report which lists all of the indicators for particular clinical entities. See Section 6 for evaluation of the output report and Section 7 for an explanation of the rules which are used by Dave.

SECTION 2. GETTING STARTED

To get the computer to read any command that you type into Dave you must hit the return or enter key. This is always the case, therefore it is not mentioned anywhere in the instructions other than here.

There will be times when more information is being

displayed to you than will fit on the screen. The screen will continue to scroll by until the end is reached. To stop the scroll temporarily, use the control key and the "s" at the same time. The space bar to start the scroll again. This is common when you are displaying lists of rules in memory or when you are adding new rules which have many hypotheses.

When you are displaying permanent rules or help screens the scrolling is different. One screen of information will display and wait for you to use the space bar to get the next screen of information. It is possible to stop the display of these screens before they reach the end by entering a "q".

Dave is designed to run using Franz Lisp under the UNIX operating system. The system for logging on to your computer under UNIX will vary for each installation. See your system manager for instructions.

In order to load Dave simply type its name in small letters.

```
dave
```

This will put you into LISP and load the part of Dave which runs the score entries and evaluation. You will now see the LISP promt which looks like this:

```
->
```

To start Dave and load the rules and menus simply type the same thing inside parentheses.

```
(dave)
```

SECTION 3. ERROR RECOVERY

Most of the time, if you make an error, you will receive an error message and be given a chance to make the entry again. However, there are some errors in which the operating system causes Dave to abort. If this happens, you will receive the Lisp promt. It looks like this:

->

If that happens, Dave must be restarted and the thing that you were doing must be redone. You should not have to redo things which were already completed. For example, if you had finished entering the WAIS scores and got an error half way through the Observation & Background Menus, then only the menus would have to be redone. Do not use the load command as you did the first time. Doing this will errase all of the work that you have done. Restart Dave using the following command:

(dave)

When Dave is loaded, the evaluation menu will appear. This menu is as follows:

EVALUATION MENU

1. Enter previous WAIS scores.
2. Enter current WAIS scores.
3. Enter local scores.
4. Enter observation & background information.
5. Perform evaluation of test results.
6. View latest report on terminal.
7. Print latest report.
8. Produce report with all rules true.
9. Teach Dave Menu.
10. Exit Dave.

There is one unlikely but possible error that you should

know about. If you are producing an output report from Item 5 of the Evaluation Menu and an unusually long period has gone by without getting the next menu then there are two possibilities. Your computer may be down or there was an error in Dave while producing the report. At this time you should appear to be locked up. Nothing that you type will appear to the screen. Try typing a carriage return or enter then use the command:

(come-back)

If you then get items printed to the screen when you type then the error was in Dave. Print the latest report and give it to a programmer.

SECTION 4. ENTERING VALUES OF SUBTEST SCORES

The first two items of the Evaluation Menu allow the user to enter the results of a current or previous WAIS test into the computer. Dave prompts the user for the results of each subtest in turn and then calculates several composite scores. These scores are initialized to zero; therefore, if they are not recorded prior to evaluation of the data, they are assumed to be zero.

Often a series of tests are administered in addition to the WAIS. In this case, the local clinical psychologist may elect to have these test results incorporated into the rules being evaluated. Item 3 is essentially the same as the first two but it works on this list of scores which have been added locally

(See Item 3 of Teach Dave Menu below). These scores are also initialized to zero.

SECTION 5. OBSERVATION & BACKGROUND MENUS

Item 4 of the Evaluation Menu presents you with ten menus which are lists of possible observations of behavior during administration of the test or known background information. You should enter the item number of each of the statements which are true for the examinee followed by a "+" . The list of menu numbers and item numbers entered are used in the evaluation. If this entry is not used before the evaluation is done, it will then be assumed that none of the menu items were true.

SECTION 6. THE REPORT

Item 5 of the Evaluation Menu causes the interpretation of the data entered in Items 1 thru 4 using the sd-qualifiers and the sign-rules (See Section 6 for explanation of the rules). A report is produced and written to a file. Item 6 will show the report on the screen. If a printed copy of the report is desired then Item 7 should be used.

During the production of the report a set of three evidence scores are determined for each clinical entity which has not been disqualified. They are, in order of importance, observation, WAIS score, and background evidence scores. Clinical entities will appear in order of the highest evidence

scores. This is Dave's way of indicating which clinical entity is most likely to be present in the examinee.

If a clinical entity has met the conditions for definate presence from the sd-qualifiers (See next section) then it will be marked as "superqualified" and the clinical entity will receive 100 points on its observation evidence score. The superqualification rule evidence and all of the sign rule evidence (See sign-rules in next section) will be listed with each clinical entity along with the hypothesis information.

The hypothesis information contains three type entries. The main entry will contain the clinical entity name and qualifier information. This qualifier may indicate the certainty of the hypothesis or it may be an adjective which adds more information to the entity. The secondary hypothesis gives a more specific diagnosis of the first hypothesis and also contains qualifier information. The additional clause in the hypothesis contains information which you will evaluate. If this information is also true then this hypothesis is more likely to be true for this entity.

Some clinical entities may also contain a section called "might add rules". This section also is a list of true evidence checks and hypotheses. These rules are intended to add further information only if this clinical entity is found to be present in the examinee.

Care must be taken to consider the three evidence scores

in relation to the maximum score possible for that clinical entity. For example, there may be twenty-seven possible rules which could cause a point for the WAIS score for psychosis but only five for mental retardation. If ten rules fire (i.e. are found to have their evidence present) for psychosis and all 5 fire for mental retardation then the score for psychosis will appear twice as strong even though every rule for mental retardation fired. For this reason Item 8 was included to provide a view of the scores for each clinical entity with every possible rule fired.

The next section in the report is "Other Observations". It also contains evidence checks and hypotheses but they do not involve one of the evaluated clinical entities. If a hypothesis for one of the clinical entities appears on this report then there is likely to be a conflict. One of the SD-Qualifiers has disqualified that clinical entity from consideration but one of the sign rules has indicated that it is possible.

The scores and the list of responses from the Observation & Background Menus are also listed on the report as a check to ensure that the proper data was used for the evaluation.

This report is then used by you to develop a plan for further testing and treatment for the patient. Refinement of Dave's rules and adaptation to suit local needs is encouraged. Facilities have been provided for a trained clinical psychologist to modify Dave without having any programming

knowledge starting with the Teach Dave Menu.

SECTION 7. THE RULES

There are two types of rules used by Dave, the sd-qualifiers and the sign-rules. There is a sd-qualifier rule for each clinical entity. Each rule has two parts. The first part checks a clinical entity for disqualification. If so, it is removed from further consideration. The second part checks for strong support of the clinical entity. If so, it is superqualified, an explanation of the supporting evidence is placed on its output report, and 100 points is added to the observation evidence score for that clinical entity.

An example of a sd-qualifier rule is as follows:

```
(neurosis (disqualifiers (and (observe 5 12)
                              (observe 6 1))
          (superqualifiers (and (observe 5 11)
                                (observe 5 13)
                                (observe 6 2))
   ("observe: History of anxiety attacks or an observed attack"
    "and  observe: Reality contact good"
    "and  observe:  Response  to  a crisis with  depression  or
                                                      anxiety")))
```

The sign-rules are used to formulate possible hypotheses if the evidence contained in the rule is present. There are three types of evidence possible in the rule, WAIS score, observation, and background evidence. All evidence listed in the rule must be present for the rule to be fired, i.e. considered in the output report. The score evidence checks patterns of current or previous WAIS subtest results and other test scores if added by the local psychologist. The observation and the background evidence checks the presence of a menu and

item number on the list of entries made by the clinical psychologist.

Each of the sign-rules may or may not be identified for the might-add list. This is information which would be of use to the clinical psychologist if that clinical entity is otherwise found to be present. It is identified by a key word "certain" and one of the main clinical entities.

A sign-rule may have one or more hypotheses. Each hypothesis has three parts, the main, the secondary, and the additional. The main part of the hypothesis specifies an entity which may or may not be one of the main clinical entities. It indicates that the condition listed is present. To determine the level of certainty that that condition or entity is present, there is a qualifier clause. It is identified by the key word "qualifier" and a string such as "probably" or "sometimes may be latent".

The secondary part of the hypothesis gives a more detailed description or more specific hypothesis than the main. Its format is the same as the main except that if its entity is "none" then the qualifier is omitted. The secondary is not evaluated but it is provided as information on the output report.

The additional part of the hypothesis gives more evidence which the clinical psychologist may examine to provide additional support for the main and secondary hypotheses. Its

format is the key word "additional" and a string description of the additional evidence. It is not evaluated but is provided as information on the output report. If this additional information is found to be true then the hypothesis will be more likely to be true.

When you are ready to add rules or change existing ones you will have to know how to put multiple pieces of evidence together using AND and OR. First a review of what they mean:

(evidence check 1 AND evidence check 2)

This is true only if both of the evidence checks are true. Now an example of OR.

(evidence check 1 OR evidence check2)

This is true if one of them are true. It is also true if both are true but it is not necessary as in the AND.

The types of evidence for the sign-rules and the sd-qualifiers is the same but the way the pieces of evidence are put into the rule is slightly different so they will be discussed seperately. The sign-rules have a list of evidence checks which is sometimes followed by a might-add flag then by the hypothesis list. Each piece of evidence listed must evaluate to be true. This is like they were all connected with the AND discussed above. However, one WAIS score check may be a very complex statement. It may be a series of WAIS score comparisons connected with AND and OR. In order to use the AND and OR statement you must surround each AND/OR clause with

parentheses. For example:

(score check 1 AND score check 2)

Each side of the AND/OR phrase may be a simple WAIS score check or it may also be an AND/OR clause. Just keep track of which expression goes with the AND/OR connector by using more parentheses. In the following example I will substitute another AND/OR clause for score check 1 above:

((score check 3 OR score check 4) AND score check 2)

When you are entering these expressions you will be asked whether the next item is a WAIS score comparrison or a left parenthesis. For the latest example you would enter two left parentheses and then enter score check 3. The right parentheses are entered for you. Make sure that you only use parentheses for this purpose. The following example would cause an error:

(score check 1) AND (score check 2) OR (score check 3)

wrong!!!

The sd-qualifiers are similar except that the observation and background evidence checks are also combined with WAIS score evidence with the AND/OR statements. For example:

(score check 1 AND

(observation check 1 OR observation check 2)

In the sd-qualifiers all of the evidence checks are combined into one statement using AND/OR for the qualifier check and the superqualifier check. However the sign-rules has a list of evidence which all must be true. Only the various

WAIS score checks are connected using AND/OR.

SECTION 8. THE TEACH DAVE MENU

The process of refining and updating Dave's menus and rules will fall primarily on you and other clinical psychologists who specialize in evaluation of intelligence test results. You have the ability to evaluate them under actual conditions and compare Dave's findings with those of the acknowledged experts in the field. With the advantage of hindsight, you may evaluate past cases whose mental condition is precisely known and compare them with Dave's evaluation of initial observations and test results. More than half of Dave's coding is devoted to its facility of allowing the clinical psychologist to improve Dave's menus and rules without calling in a LISP programmer for a rewrite. It also allows you to incorporate other test scores into the rules based upon further research or local policy. The Teach Dave Menu is used in order to make changes to the rules, the local scores, the clinical entities, or the observation and background menus. When Item 9 of the Evaluation Menu is executed the following menu will appear:

TEACH DAVE MENU

1. Change observation & background menus.
2. Change the clinical entities to be examined.
3. Change the local score names to be examined.
4. Change sign rules.
5. Change Superqualifier/Disqualifier rules.
6. Make file changes permanent.
7. Return to main evaluation menu.
8. Exit Dave.

Item 1 is provided in order for the user to add or change observation and background entries to the menus in Item 4 of the Evaluation Menu.

Currently Dave provides a wide range of observations but only evaluates, scores, and produces a report for seven clinical entities. They are neurosis, psychosis, antisocial personality disorder, organic brain syndrome, mental retardation, and giftedness. If, in the process of evaluation of Dave, it is found that it would be desirable to add other entities on this same level such as depression, anxiety, or schizophrenia, then they may be added using Item 2. Clinical entities may also be deleted using this facility.

It is essential to be able to incorporate the results of other tests into this evauation, either as part of the refinement process or to suit local conditions. Item 3 is the means by which new score names are entered into the system. You will be asked for a description of the score subtest and a one word abbreviation of the score subtest name. The program will perform a check to ensure that the abbreviated name does not conflict with other names used elsewhere. Rules governing the evaluation of these new data items should then be added to the sign-rules and/or sd-qualifiers using Items 4 and 5 below. The results of these new data items will be entered using item 3 of the Evaluation Menu. (See item 6 below).

Items 4 and 5 are used to change the rules which are evaluated by Dave to produce the report in Item 5 of the

Evaluation Menu. If a new clinical entity is added, it is then likely that a sd-qualifier will be added for it, giving conditions for superqualification and disqualification. Also additional sign-rules or changes to existing sign-rules should be made for these new entities. Items 4 and 5 are the primary tools of the user in the refinement of Dave's set of rules.

After a rule has been added you will be presented with a listing of that rule in LISP code. You will be asked if it is correct. Do not be alarmed if it looks strange. Be impressed at your ability to write such a complicated piece of code. The WAIS score comparisons will be in the reverse order of the way you entered them. (score check 1 and score check 2) will appear as (and score check 1 score check 2). Also there will be parentheses all over the place. If there are no serious misspellings and everything appears to be grouped together properly then it is likely that the rule is correct. When you indicate that the rule is correct then it will be added to the rules in memory.

In Items 1 thru 5 of the Teach Dave Menu, the changes made are only made in the current computer memory and not to the permanent file. This is desirable during the process of refinement because changes can be made without fear of damaging the permanent rules. Experiments can be carried out with different combinations of rules and/or clinical entities to see what the change in the results will be. After you have produced a report which you are satisfied is better than the one you had under the permanent rules, you may choose to make the changes

permanent using Item 6. You will be presented another menu asking which one of the sets of rules, scores, or clinical entities will be made permanent. Each of these rules, scores or clinical entities is a list which is maintained in a separate file. The process of making these lists permanent involves replacing the old file with one which reflects what is currently in memory. Warning: If changes are made using Items 1 thru 5 above, and item 6 is not used then the changes will be forgotten when LISP is exited or Dave is reloaded.

APPENDIX R, CHANGE SIGN-RULE ALGORITHM

This section describes the algorithm for adding more sign-rules or modifying existing ones. It uses terms which are introduced in the grammar of the rule discussed in Chapter 3, Rule Design. This aglorithm does not reflect all of the detail that actually went into the code. Actually the user is offered the opportunity to re-do portions of rules, start over, get a help screen presented, and several other options. This section is intended to present an overview of the process. See Appendix O for the documented code.

This algorithm is highly recursive and functional, i.e. the significant values are returned in the values of the function. In some cases the names of the paragraphs match functions in the actual code but the description may include properties of functions which are performed by other functions called by it.

change-rule select one:

* display permanent sign-rules
* display sign-rules in memory
* change-current-rule
* add-rule

change-current-rule

* Starting at a user prescribed rule number, display rules until the user indicates to change the rule.
* When target rule is selected, put sign-rules back together with all rules before the target + get-substitute-rule + all rules after target.

```
add-rule

    *  form new rule with "rule" + last rule number plus one  +
       get-rule-evidence + get-hypothesis

    * add new rule to the end of sign-rules

get-substitute-rule

    *  form  new rule with "rule" + target rule number  +  get-
      rule-evidence + get-hypothesis.

get-rule-evidence   case user-selection of evidence type:

    * WAIS score // get-score-rule + get-rule-evidence

    * Observation // get-observe-evidence + get-rule-evidence

    * Background // get-bckgrnd-evidence + get-rule-evidence

    * Lack of background // "not" + get-backgrnd-evidence +
                                     get-rule-evidence

    * Lack of observation // "not" + get-observe-evidence +
                                      get-rule-evidence

    * mignt-add-flag // get-certain

    * end of evidence // nil

get-observe-evidence

    * get menu number and item number

    * if not valid then get-observe-evidence
      else "observe" + menu number + item number

get-bckgrnd-evidence

    * same as observe except key word "bckgrnd"

get-certain

    * get clinical entity name

    * if clinical entity name valid in clinical-entities list
      then "certain" + clinical entity name

get-score-evidence  case user-selection of evidence complexity

    * WAIS score // get-relative-operator

    * Left parenthesis // get-logical-operator
```

```
get-logical-operator

    * first = get-score-evidence

    * case user selection of logical operators:

        ** AND // "and" + get-score-evidence + first

        ** OR // "or" + get-score-evidence + first

get-relative-operator

    * first = get-arithmetic-operator

    * get relative operator from user

    * relative operator + first + get-arithmetic-operator

get-arithmetic-operator

    * first = get-item

    * get arithmetic operator from user

    * if arithmetic operator = "none"
      then first
      else
        arithmetic operator + first + get-arithmetic-operator


get-item

    * get data item from user

    * if data item is a number or valid score identifier
      then return data item

get-score-description

    * read line of text as a string

    * if line of text = " +"
      then nil
      else line of text + get-score-description

get-hypotheses case user selection  of final hypothesis:

    *  y  //  "hypothesis" + get-main +  get-secondary  +
              get-additional

    *  n  //  "hypothesis" + get-main +  get-secondary  +
            get-additional + get-hypotheses
```

```
get-main

    * get entity from user

    * "main" + get-qualifier + entity

get-secondary

    * get entity from user

    * if entity + "none"
      then "secondary" + entity
      else "secondary" = get-qualifier + entity

get-additional

    * read line of text as a string

    * "additional" + line of text

get-qualifier

    * read line of text as a string

    * "qualifier" + line of text
```

APPENDIX S, CHANGE SD-QUALIFIER

This section describes the algorithm for adding more sd-qualifier rules or modifying existing ones. It uses terms which are introduced in the grammar of the rule discussed in Chapter 3, Rule Design. This section is actually a continuation of Appendix R in that it performs the same function except on a diffenent set of rules. The grammars of both types of rule are found in Chapter 3, Rule Design. Some of the functions described below use functions in the previous appendix and are discribed in the same format. Functions which are described in the previous appendix are surrounded by square brackets. The functions are highly recursive and the significant values are returned in the value of the function.

change-sd-qualifier select one:

* display permanent sd-qualifiers from file
* display sd-qualifiers currently in memory
* change-sd-qualifier
* add-sd-qualifier

change-sd-qualifier

* display each rule in turn until user indicates that it is target for change.
* sd-qualifiers = all rules before target + get-substitute-sd-qualifier + all rules after target

get-substitute-sd-qualifier

* clinical-entity = clinical-entity of target
* clinical-entity + "disqualifiers" + get-disqualifier + "superqualifiers" + get-superqualifier

```
add-sd-qualifier

     * get valid clinical-entity which does not have current sd-
       qualifier from user

     *  new  rule  =  clinical-entity +  "disqualifier"  +  get-
          disqualifier + "superqualifier" + get-superqualifier

 get-disqualifer

     * get-sdq-evidence

 get-superqualifier

     * get-sdq-evidence + [get-score-description]

 get-sdq-evidence  case user selection of evidence type:

     * Left parenthesis // get-sdq-locical-operator

     * WAIS score // [get-relative-operator]

     * Observation // [get-observe-evidence]

     * Background // [get-bckgrnd-evidence]

     * Absence of Observation // "not" + [get-observe-evidence]

     * Absence of Background // "not" + [get-bckgrnd-evidence]

 get-sdq-logical-operator

     * first = get-sdq-evidence

     * case user selection of :

          ** AND // "and + get-sdq-evidence + first

          ** OR // "or" + get-sdq-evidence + first
```

APPENDIX T, INFERENCE ENGINE ALGORITHM

This procedure is executed by Item 5 of the Evaluation Menu. It uses the values associated with the three lists of identifiers which are assigned in Items 1 thru 3 of the same menu. They are the local-score-list, previous-score-list, and score-list. It also uses obs-list which is assigned value by Item 4. Obs-list is a list of all of the entries from the Observation & Background Menus. A sample element of this list is as follows: (8 2 " Obvious presence of Down's Syndrome"). Evaluation of Observe and Background evidence consists of checking this list to see if the menu and item number are present.

These procedures create the lists which will be used to print the output file. Each clinical entity name is used as an identifier which has a value of all of the rules which are fired and have a hypothesis which indicates the clinical entity in the main clause. If a fired rule doesn't match one of the fired clinical entities, it is added to the other-fired-rules list. Each clinical entity has three evidence scores associated with it which are used to judge the most probable entities present for this patient. The identifier associated with each of these scores is the concatenation of the clinical entity name and the words "observe", "score", or "bckgrnd".

Each clinical entity also has a list associated with it for any rules which may be identified for the might-add lists. This list has as its identifier the clinical entity name concatenated with the word "might-add".

The first step in producing these scores and lists is to evaluate the sd-qualifiers. Fired-clinical-entities is the working list of clinical entities. A copy of the clinical entities is made because the entire list is needed in other places in Dave and this procedure removes those which are being disqualified.

```
fired-clinical-entities = clinical-entities
initialize the scores mentioned above to zero
initialize the other lists mentioned above to nil
evaluate all sd-qualifier rules
   * evaluate disqualifier evidence
         if  true then remove clinical entity from
                                  fired-clinical-entities
   * evaluate superqualifier evidence
       if true then add "superqualifier" and superqualifier
         evidence description to clinical entity identifier
evaluate each sign-rule
   * fired-rule = nil
   * points = true
   * evaluate score, observe, and bckgrnd evidence
         if all evidence true then add
                          "score" + score descriptions
                          or observe clause
                          or bckgrnd clause to fired-rule
                    else go to next rule

   * if certain clause is present
        then add fired-rule and following hypothesis
              to the clinical-entity-might-add identifier

         otherwise
           add fired-rule and each remaining hypothesis
```

```
    to the clinical-entity identifier indicated
    in the main clause of the hypothesis.
AND
for each rule in this category do the following
 if  points = true and main hypothesis clinical
 entity is contained in fired-clinical-entities
  then  increment  by 1 the score identified by
  concatenation of clinical entity and key word
  of the first evidence of the rule
  points = false (only first matched hypothesis
                  counts for a evidence score)
```

Sort the clinical entity output lists according to the evidence scores described above. Observation evidence is considered the most reliable followed by WAIS score evidence and Background is leat reliable. Determine the order of output by a composite score obtained by

```
Composite-score = Observation score * 100 +
                  WAIS score * 10 +
                  Background score .
```

Produce report with fired-clinical-entities sorted in order of Composite-score and written to a file so that it can be printed to the screen or the printer at the request of the user. The steps are as follows:

* print title
* print each fired-clinical-entity with rules fired for it
  - print name of clinical entity
  - print description of each rule fired
  - print might-add rules if any
* print description of rules in other-fired-rules list
* print scores and menu selections used in evaluation

IMPLEMENTATION OF DAVE: AN EXPERT SYSTEM FOR THE ANALYSIS OF THE WECHSLER ADULT INTELLIGENCE SCALES AND RELATED INFORMATION

by

GLEN ROSS WHITE

B. S., Mathematics, Louisiana State University, 1972

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

Kansas State University
Manhattan, Kansas

1985

ABSTRACT

This report describes the implementation of Dave: an expert system for the anaysis of the Wechler Adult Intelligence Scales (WAIS) and related information. The WAIS was developed as a measure of intelligence. It is what most people know of as the IQ test. However, clinical psychologists have been using them for many years to assist in diagnosing other clinical entities. The ones considered by Dave include psychosis, neurosis, antisocial personality disorder, organic brain syndrome, mental retardation, and giftedness. Researchers have found that certain subtest scores on the WAIS will give an indicator that one of these clinical entities may be present. This evaluation occurs manually by the clinical psychologist and requires approximately two hours of work. The quality of the evaluation depends upon the training of the psychologist and upon his ability to apply the appropriate rules. Dave produces a report which contains English language descriptions of all of the evidence which indicates the presence of various clinical entities. The clinical psychologist will use this report to develop a plan for further testing and therapy.

Dave provides a comprehensive facility for user modification of rules, observation/background menus, and clinical entities. It also allows for the user to develop rules for the evaluation of other test scores in addition to the WAIS.

The paper discusses the design and implementation of the

rules and the functions which evaluate them. It introduces the theories which led to the development of the two types of rules. The rule structure and purpose is explained and the major functional algorithms are listed. The documented code is included with sample output and a users manual. Finally, Dave's limitations and the areas which require further work are discussed and conclusions are made.