83-29

Reports

of the

ISG

PLANT/ds:
An Expert System for the Diagnosis of Soybean Diseases Common in Illinois

User's Guide and Program Description

by

Robert Reinke

October 1983

File No. UIUCDCS-F-83-912

ISG 83-~~12~~

File No. UIUCDCS-F-83-912

PLANT/ds:

An Expert System for the Diagnosis of Soybean Diseases Common in Illinois

User's Guide and Program Description

by

Robert Reinke
Department of Computer Science
University of Illinois
Urbana, Illinois

October, 1983

ISG 83-12

This work was supported in part by the Office of Naval Research under Grant No. N00014-82-K-0186 and in part by the United States Department of Agriculture under Grant No. 321512344.

PLANT/ds
USER'S GUIDE

1. Introduction

PLANT/ds is an expert system for diagnosing soybean diseases common in Illinois. The system makes diagnoses on the basis of the answers to specific questions about the diseased crop and its environment. Questions are presented to the user in a simple tabular format designed to increase bandwidth. The system provides easy access to already answered questions, allowing the user to correct answers and experiment with the effects of various answers on the diagnosis. Because of this, PLANT/ds can also serve as a tutorial system.

This report describes the latest version of the soybean disease diagnosis system described in [Michalski and Chilausky 1980] and [Michalski et. al. 1982]. PLANT/ds was developed in the context of the ADVISE "meta-expert" system [Michalski et. al. 1983, Michalski and Baskin 1983]. ADVISE runs under UNIX on a VAX mainframe, and has been used to develop several expert systems [Boulanger 1983, Rodewald 1983]. PLANT/ds on the IBM PC is somewhat independent of the ADVISE system due to the difficulty of implementation on a personal computer, but a version of PLANT/ds identical to the one described here does run on the VAX under ADVISE.

The PLANT/ds user interface was carefully designed, as the system is intended for use by farmers and plant pathologists. Those with little or no experience with computers should be able to use the system effectively after skimming sections 3 and 4 of this guide. For those who desire more detailed information about PLANT/ds, Section 2 contains a discussion, in general terms, of how the knowledge base and inference mechanism in PLANT/ds work. Programmers who wish to modify or extend PLANT/ds should refer to the document "PLANT/ds Program Description."

2. The Knowledge Base and Inference Mechanism

The knowledge base in PLANT/ds consists of a set of *decision rules*, one decision rule for each disease the system knows about. A rule states the symptoms associated with a disease, and the importance of each symptom in the diagnosis of that disease. The decision rules were obtained by standard knowledge engineering techniques (i.e. by questioning plant pathologists about soybean disease diagnosis). Earlier versions of the program included rules derived through inductive inference [Michalski and Chilausky 1980]. These rules, though excellent at batch diagnosis of large numbers of cases, proved too unwieldy for use in a small interactive system. Work is currently underway to improve the machine-derived rules. Once this is done, PLANT/ds will contain two rule groups for use in diagnosis.

The next three subsections describe in more detail how rules in the knowledge base

are evaluated and how the inference mechanism selects rules and questions.

2.1 Decision Rules

Figure 1 shows the decision rule for Diaporthe Stem Canker. When PLANT/ds obtains some information (from the user) that applies to Diaporthe Stem Canker (e.g. the user tells the system that stem cankers are absent), the system evaluates the rule using the new information. When a rule is evaluated, a value between 0 and 1 is returned; this value is a *confidence* in the belief that the given disease is present. Confidence is a relative measure expressing the degree to which the system believes a disease is present; it is *not* a probability. If the degree of belief falls below 0.55, the disease is rejected.

A decision rule associates a number with each symptom (see Figure 1). This is the confidence PLANT/ds would have in the presence of the given disease if *only* that symptom were present. The confidence in a disease is calculated by taking the probabilistic sum of the weights associated with each symptom. The weight associated with a symptom in this calculation is the number that appears next to that symptom in the decision rule *unless the the symptom is not present.* In this case, the weight associated with the symptom will be zero.

To clarify these ideas, Figure 2 shows a sample calculation for the rule in Figure 1. Note that if a value is unknown (in the case shown, it is unknown whether premature defoliation is present), the full weight is used.

Decision rules are written in a subset of the variable-valued logic system VL$_1$ [Michalski 1974]. Rules may be considerably more complex than the rule shown in Figure 1. The parser used to generate the rules can also deal with conjunction and disjunction. In PLANT/ds, conjunction is evaluated as minimum and disjunction as maximum.

2.2 Restriction Rules

A second kind of rule used by PLANT/ds is the restriction rule. Restriction rules (so called because they restrict the number of questions the system will ask) are rules used to assign answers to questions without having to ask the user. One such rule in PLANT/ds is:

```
[Stem Cankers = Absent] ::>
[Canker Lesion Color = Does Not Apply]
[Location of Stem Cankers = Does Not Apply]
```

These rules are used to make common-sense deductions about the condition of the crop and avoid asking the user unnecessary questions.

2.3 The Inference Mechanism

PLANT/ds does not use either of the inference mechanisms (forward chaining or backward chaining) commonly associated with rule-based systems. Instead, the selection of a question is based on that question's utility. The definition of utility changes through the consultation, as described below. The selection of a rule to fire

is based on the question selection (i.e. the only rules fired are those whose status may have been affected by the answers just received).

When the consultation begins, PLANT/ds treats all of the diseases it knows about as *working hypotheses*. PLANT/ds tries to select questions in such a way that the maximum number of hypotheses will be rejected at each step.

The first few questions asked by the system are "standard" questions selected beforehand. These provide information that helps PLANT/ds determine, in general, what kind of disease is present. The questions attempt to eliminate those diseases that could not possibly be the cause of the problem. For example, one of the first questions will inform the system whether the leaves are abnormal. If they are normal, PLANT/ds will be able to reject any disease which affects only the leaves.

Once these standard questions have been answered, a different strategy is adopted. If there are more than five hypotheses remaining, the system chooses questions that are relevant to the largest number of remaining diseases. This step is repeated until there are fewer than five hypotheses remaining (or until all relevant questions have been answered).

When fewer than five working hypotheses remain, the control scheme again changes strategy. It selects the disease with the highest confidence at present and asks questions pertaining to that disease. This step is repeated until either all unanswered questions have been answered or until all the remaining hypotheses are eliminated.

When all the questions relevant to the remaining hypotheses have been answered, PLANT/ds is ready to give a diagnosis. There may be more than one disease given in the diagnosis, but this is usually because symptoms for many of the diseases overlap. There will generally be one disease with a confidence clearly above the others; this should be regarded as PLANT's primary diagnosis.

3. Starting PLANT/ds

PLANT/ds is available on two 5 1/4" floppy disks for the IBM Personal Computer. The program needs 128K of RAM on the IBM PC, and will run on a standard 80 x 24 black and white monitor.

To start PLANT/ds on the IBM PC, put the diskette labelled "PLANT : #4" in the left hand drive (with the label up), and the "PLANT : #5" disk in the right hand drive, then bootstrap the PC. When the IBM logo appears on the screen, type "x". This will cause the prompt "Execute what file?" to appear on the screen. Type "#5:plant" and hit RETURN. Nothing will happen for approximately one minute (the PLANT/ds system requires this time to prepare), then the consultation will begin.

To start PLANT/ds on the University of Illinois Department of Computer Science vaxb, execute the following commands:

1) cd /mntb/2/michalski/isg/PLANTds
2) plant

4. Using PLANT/ds

Much of the material in this section is contained in the help facility built into PLANT/ds (see subsection 4.3.1). The material here is slightly more detailed, and should be referred to if any questions arise.

4.1 General Comments and Overview

PLANT/ds was designed with ease of use in mind. With one exception (see section 4.3.4), all commands are executed with a single keystroke. Instructions always appear on the screen towards the bottom, and information about the user's location within the system always appears at the top of the screen.

Three keys are very important in operating PLANT/ds. The first is the RETURN key. RETURN always means the same thing – it tells PLANT/ds that the user is done looking at the information currently on the screen and wants to proceed (on the IBM PC, the RETURN key is located on the right edge of the main keyboard and is labelled with an arrow pointing down and to the left; this key is called ENTER in the IBM PC user's manuals). The other important keys are SPACEBAR and BACKSPACE. These keys are used to move the cursor forward and backwards in some parts of the system (see section 4.2). The BACKSPACE key is directly above the RETURN key on the keyboard.

PLANT/ds has two major parts: the question forms (section 4.2) and the options page (section 4.3). Figure 3 shows how these parts are linked together.

4.2 Question Forms

Question forms are the means through which PLANT/ds gets information about a crop problem. These forms are an electronic version of standard paper and pencil forms; the user simply puts an "x" in the box next to the appropriate answer. Figure 4 shows a typical question form (this is the form that appears at the start of every consultation). The header line at the top of the form gives a question form number and a label for the form. These are useful if the user decides he wants to change the answers on the form at a later time (see section 4.3.5).

As soon as one question on the form has been answered, the cursor automatically jumps to the next question. Similarly, when all the questions on one form have been answered, PLANT/ds automatically proceeds to the next. The user need not answer every question on a form; if he does not know the answer, the question can be skipped. If some questions are skipped the user must press RETURN to tell the system that he is done with the form on the screen.

When a question form is completed, PLANT/ds will clear the screen and begin *testing hypotheses*. This means that the system is evaluating rules (see section 2.1) to see if the answers on the last form changed the status of any hypothesis. Testing hypotheses may take up to a minute; if during this time the user realizes that he gave an incorrect answer on the previous question form, he can press the ESC key. This causes PLANT/ds to return to the previous form immediately. If hypothesis testing is not interrupted, the system will proceed to the next question form automatically.

[Canker Lesion Color = Brown : 0.45]
[External Stem Discoloration = Brown : 0.40]
[Time of Occurrence >= July : 0.15]
[Premature Defoliation = Present : 0.10]

::> [Soybean Disease = Diaporthe Stem Canker]

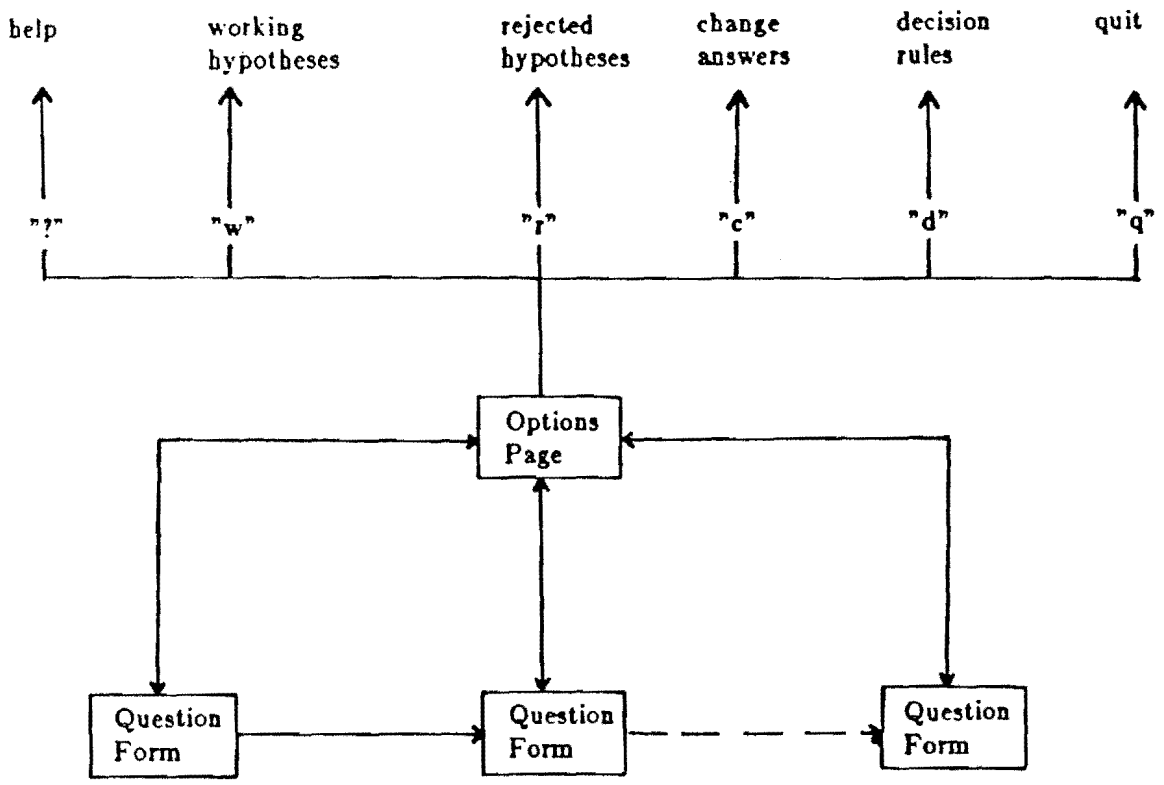
Decision Rule for Diaporthe Stem Canker. See text and Figure 2 for an explanation of the numbers associated with each symptom.

Figure 1.

Value Given By User	*Confidence Computation*
Canker Lesion Color = Brown	0.45
External Stem Discoloration = Absent	0.00
Time of Occurrence = August	0.15
Premature Defoliation = Unknown	0.10
Value for Rule (psum) =	0.59

Sample calculation for the decision rule in Figure 1. See text for further details.

Figure 2.



A Diagram of the PLANT/ds system. See section 4 for a detailed discussion of each of the parts shown.

Figure 3.

Question Form 1: Diseased Areas

Fruit Pods	( ) Normal	( ) Abnormal
Seeds	( ) Normal	( ) Abnormal
Roots	( ) Normal	( ) Abnormal
Leaves	( ) Normal	( ) Abnormal
Stem	( ) Normal	( ) Abnormal

"x" – make an entry
"e" – erase an entry
"o" – see other available options
"?" – get help for this question

RETURN – leave this screen of questions
SPACE – go forward to the next entry
BACKSPACE – go back to the previous entry

A typical question form. The top line gives a title to the form; forms that appear later in the consultation will also contain information as to the number of remaining hypotheses.

Figure 4.

PLANT/ds: Options Page

? – get information about using PLANT/ds
w – see the working hypotheses
r – see the rejected hypotheses
d – display some decision rules
c – change the answers on an earlier question form
q – quit this session (all information will be lost)
RETURN – return to the consultation

Figure 5.

Most of the commands available on the question forms have to do with moving the cursor about and answering questions, but there are two other commands that are very important. The first, "?", will present a short paragraph explaining the current question and how it should be answered.

The second important command, "o", allows the user to access the *options page*. The options page gives access to a large amount of information about how the consultation is going and why PLANT/ds is making the decisions it is. The options page is discussed in detail in the next section.

4.3 The Options Page

The options page is accessed by typing "o" on any question form. It is also available, in modified form, at the end of the consultation. Figure 5 shows the options page as it appears on the screen.

4.3.1 Getting Help

As mentioned earlier, PLANT/ds contains an interactive help facility. This is accessed through the "?" option on the options page. Much of the information in this manual is duplicated in the help facility. The use of the help facility should be self-explanatory.

4.3.2 Working Hypotheses

Typing "w" on the options page allows the user to see those hypotheses (diseases) that the system still considers as candidates for the cause of the crop problem. Two numbers are associated with each disease; the *maximum possible confidence* and the *estimated final confidence*. The maximum possible confidence is the number obtained by evaluating the rule using the method discussed in section 2.1. The estimated final confidence is computed in the same way, except that if a condition is unknown, only half of the associated weight is used in computing the value of the rule.

4.3.3 Rejected Hypotheses

Typing "r" on the options page allows the user to see those diseases that the system has rejected as possible causes of the crop problem. A hypothesis is rejected when its maximum possible confidence degree falls below 0.55. The two numbers associated with each disease are computed in the same manner as for working hypotheses, above.

4.3.4 Displaying Decision Rules

Expert systems are unique in that they allow the user to not only get an answer, but allow him to see how that answer was arrived at. The "d" option on the options page allows the user of PLANT/ds to see, in understandable form, the decision rules PLANT/ds uses to make its diagnosis. Decision rules are displayed in the format shown in Figure 1. Section 2.1 contains a

discussion of how these rules should be interpreted.

4.3.5 Modifying Answers

At any time, it is possible to go back and change the answers on a question form that was completed some steps earlier. The "c" option on the option page presents the user with a list of question forms that have been completed, and allows him to chose one to go back to. The standard forms (section 4.2) are identified by the labels that appear at the top of them. The remainder are identified by the questions that appear in them. Choosing an old question form is the only time the user must use more than one keystroke to execute a command. Completed question forms are numbered; to see a form, the user types the number followed by RETURN.

PLANT/ds treats old question forms differently from new forms in two ways: first, a form being modified always has the label "Modifying Answers;" second, the user must always press RETURN to tell the system when she/he is done making changes.

When the user is done modifying the old question forms, PLANT/ds will test its hypotheses to see how the changes have affected the state of the consultation. Once this is done, the user can select another form to modify, or RETURN to the options page.

4.3.6 Quit

The user may, at any time, quit the consultation by selecting option "q" on the options page. If the user quits the consultation, all the information that was entered will be lost, and cannot be recovered. Because of this, the user is warned, once the "q" option has been selected, that this action will irrecoverably destroy all the answers entered up to this time. The user is then given the option of going back to the consultation.

4.4 The Diagnosis

Eventually, PLANT/ds will run out of questions about the remaining hypotheses. When this point has been reached, the system will test its hypotheses one more time, then give the diagnosis. Those diseases whose maximum possible confidences are still over 0.55 will be shown, in order of decreasing confidence, as the diagnosis.

It is possible that PLANT/ds will reject *all* the hypotheses. In this case, there are three possibilities:

1) The problem is caused by a disease PLANT/ds does not know about.

2) The symptoms were not specified correctly by the user.

3) The disease is not advanced enough for PLANT/ds to recognize the symptoms.

If you are certain that you identified the symptoms correctly, and specified them correctly to PLANT/ds, the only recourse is to consult a human soybean specialist.

4.5 Control Information

Once the user is done looking at the diagnosis, PLANT/ds presents a Final Options Page. This options page is identical to the normal options page except for two things:

1) The user may return to the diagnosis by typing "r"

2) The user may get some simple suggestions for controlling the diagnosed problems by typing "s".

The control information PLANT/ds offers is simple and general. In reality, control is a difficult problem, and methods have to be tailored to the individual field and farmer. What PLANT/ds offers in this area is simply an idea of what kind of measures might be necessary. The user should consult a soybean expert before proceeding with control measures.

PLANT/ds
PROGRAM DESCRIPTION

Version 3.2ibm

1. Introduction

PLANT/ds on the IBM Personal Computer consists of approximately 5,000 lines of UCSD Pascal code. The program requires a minimum of 128K of RAM on the PC and will run on a standard 80 x 24 black and white monitor. The code is divided into seven different modules (UCSD units) to allow separate compilation. Many of the routines are "segmented" to allow the UCSD version of page faulting to be used on them. This results in memory savings at the expense of increased disk access.

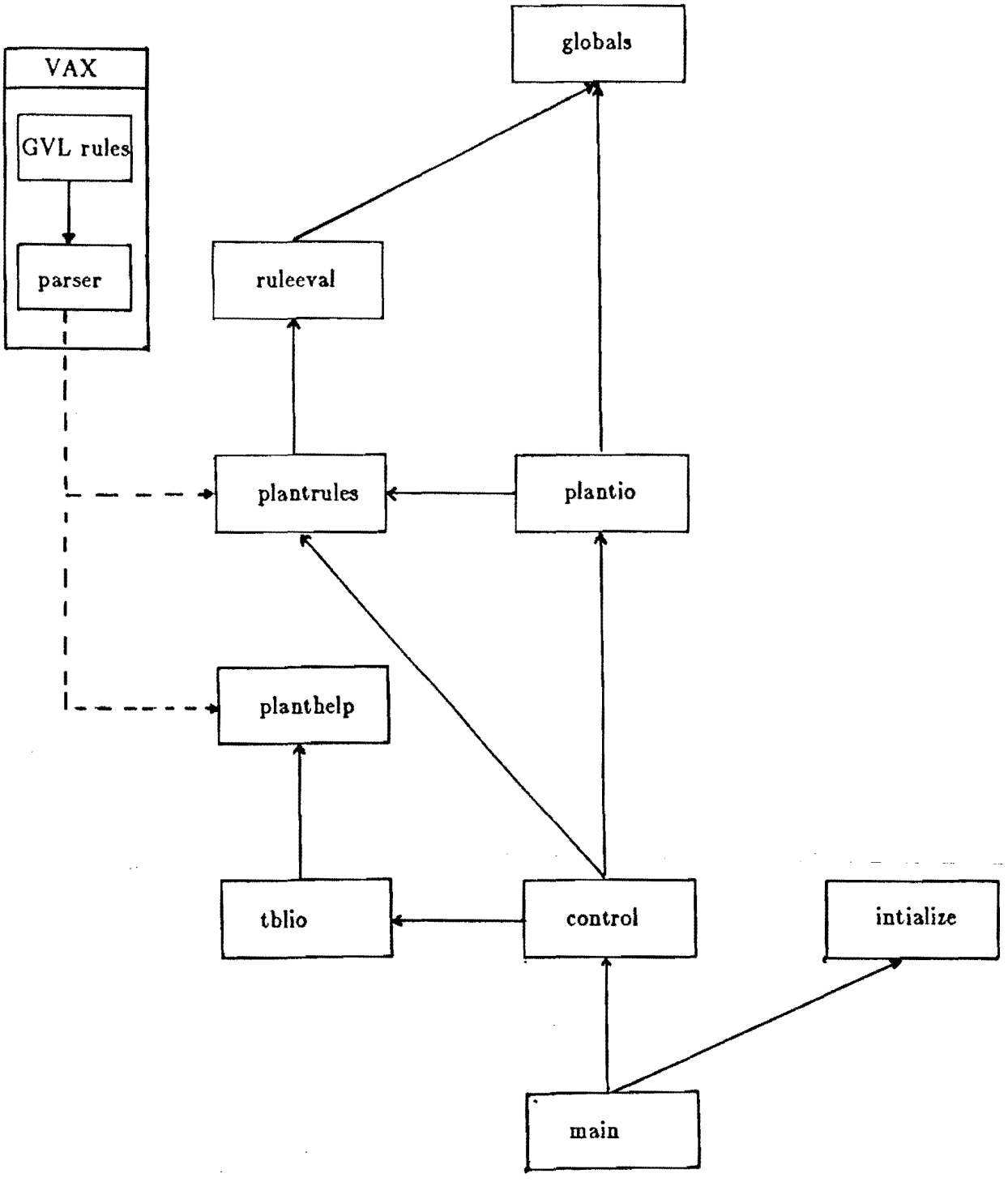
The PC version of PLANT/ds was written in the context of the ADVISE meta-expert system being developed by the Intelligent Systems Group at the University of Illinois. Although in many ways this program is a separate entity from ADVISE (due to the difficulty of implementation on a personal computer), one very important link with the larger system is maintained: the ADVISE parser has been modified (by Carl Uhrik) to produce rules executable by the PC version of PLANT/ds. In other words, the coding of a new knowledge base for PLANT/ds on the PC has been automated; once rules are written in parser input format, a new "rules" unit (see section 4.3) is automatically produced. This must then be ported (ADVISE runs under UNIX on a VAX 11/780) to the PC and can be executed immediately. However, changes in descriptors (variables) used in the rules necessitate manual changes to the PLANT/ds code on the personal computer; this process should also be automated in the future.

This document is intended to be an aid to programmers wishing to modify or extend the PLANT/ds code; familiarity with the use of the PLANT/ds system and the UCSD Pascal system on the PC is assumed. Those unfamiliar with PLANT/ds are advised to read the document "PLANT/ds User's Guide" before attempting the material in the Program Description.

2. Overview of the System

Figure 1 shows the structure of the PLANT/ds software. Each of the modules shown (except for the parser) corresponds to a UCSD compilation unit. Naturally, the modules are also divided functionally. The calling hierarchy is as shown; modules towards the bottom of the diagram *use* (in the UCSD sense) modules above them. Section 4 presents a detailed description of each of the modules shown here.

The main program module does very little of the system work, calling *initialize* and *plantctrl* to handle the initialization and the question-answering phases of the consultation, respectively.



A diagram of the PLANT/ds software modules and their relation to the ADVISE parser.

Figure 1.

The init module initializes the global variables and sets up the linked list structure of the crop descriptors (see next section). This module also reads in data about the rules produced by the parser.

The plantctrl module is the heart of the PLANT/ds system; the control scheme is implemented here. Plantctrl decides what questions should be asked, and makes the appropriate calls to have them presented. It also determines which rules should be fired.

The *plantrules* module is produced by the ADVISE parser. Rules consist of calls to the routines in the *ruleeval* module.

The *tblio* module contains routines to perform table-driven interaction with the user during a consultation. The routines in this module access a set of 10 data files which contain the text of the questions. The answers received are stored in global data structures (next section).

The *planthelp* module is also produced by the ADVISE parser. The help module consists of one routine which prints the help paragraph for the variable passed as a parameter.

The *plantio* module contains the routines for the options page and the diagnosis.

3. Global Data Structures

There is really only one important global data type in the PLANT/ds system; this is the *varstructure* type, declared as follows:

```
variable = ^varstructure;

varstructure = record
                 val       :   hypertype;
                 proped    :   boolean;
                 num       :   varnumbers;
                 possvals  :   superset;
                 expert    :   ruleset;
                 machine   :   ruleset;
                 link      :   variable;
                 semantic  :   variable;
                 lostvars  :   variable;
               end;
```

This type is used to represent crop descriptors. The *val* field of this record contains the current value of the variable described. The type *hypertype* used here is a scalar consisting of all possible values any crop descriptor could take on. This type is somewhat unfortunate; it would be nicer if each descriptor took on only the values appropriate to it. However, there is no way to declare types of this sort in Pascal (since several descriptor ranges intersect). The *possvals* field is a partial attempt to rectify this problem; this field contains the legal values for the descriptor (*superset* is declared as *set of hypertype*). The possvals are assigned in the initialization routines. They are not rigorously used during program execution, however. This is the only area in which the

program is somewhat brittle; descriptor assignments must be done very carefully in order to avoid confusing assignments.

The *proped* field in the varstructure record is set to false unless the variable described has been propagated through the rules. The *num* field contains a unique internal name for the variable; the control scheme and table driver always refers to variables by this integer. The two sets *machine* and *expert* are sets of valid rule numbers. Contained in these sets are the numbers of the rules that this variable appears in. This information is obtained at initialization time from data files produced by the parser (see section 4.1). The sets corresspond to two separate rule groups. Elsewhere in the system, these two sets of rules are refered to as rule groups 1 (expert) and 2 (machine).

The three pointer fields in this type are used to maintain three separate linked lists of variables. The first type, *link*, is used for the list of variables whose values are not known (in some sense, this is the "main list" of variables). The variables are initialized into this list in *varnumber* order, and the global variable *notknown* is set to point at the first variable. Variables are removed from the notknown list when values are entered for them by the user or by restriction rules. The second pointer field, *semantic*, is used to link together variables that will be asked for in one question form. The table-driver routines use this pointer extensively. The last pointer, *lostvars*, is related to the use of restriction rules. When a restriction rule is applied, the variables whose values become known are attached through *lostvars* links to the variable which caused their values to become known. For example, if the restriction rule:

```
[stem_cankers = absent] ::>
[canker_lesion_color = does_not_apply]
[location_of_stem_cankers = does_not_apply]
```

were to be used (i.e. stem cankers are known to be absent), then the records corressponding to canker lesion color and location of stem cankers would be linked, via *lostvars* to the variable corressponding to stem cankers.

4. Modules

This section covers the PLANT/ds modules in detail. For each module, the important variables (if any) will be presented, followed by a discussion of the major procedures and their tasks. Lastly, the files used by the module (if any) will be discussed.

4.1 Unit INITIALIZE

The init module intializes the program variables and reads in data about the rules the program will be using. There are no variables declared for the init module.

The init module is accessed by *main* through the interface procedure *init*. The first part of this procedure consists of calls to the Pascal dynamic variable allocation function *new*; these calls set up the linked list of domain descriptors. Once variables are established, procedure readrules is called. This procedure reads the file *rulevars.text*, which contains listings of the variables which will be found in each

rule. Once the data in this file has been placed into the varstruc structures, procedure slink is called to link together those descriptors which will appear in the same question form. Data about these links is in the file *slinks.text*, which must be entered by the programmer.

The file rulevars.text has one line for each decision rule the system uses. If more than one rule group is being used, the lines for rule group one must precede the lines for rule group two. The two rule groups are separated by a line with the single character '0' on it. The system expects *numrules* (a global constant) rules in each rule group. The lines are in the following format:

<rule number> <number of variables in this rule> <list of variable numbers>

The three parts of each line may be separated by any non-integer character. Similarly, the variable list for each rule contains the descriptor numbers separated from each other by non-numeric characters. Procedure readrules does not care if the variable listings for a rule are split over more than one line.

The file slinks.text contains any number of lines in the following format:

<number of first variable in question form> <number of variables in form> <list of variable numbers in question form>

Again, individual numbers are separated by any non-numeric character.

Neither of the procedures described above (readrules and slink) has any error detection or correction facilities; the data in rulevars and slink *must* in the correct format or a runtime error will result.

4.2 Unit RULEEVAL

This module contains routines that are accessed *only* by the rule modules. Four interface procedures are provided: *addmod, andresult, orresult* and *max.*

The rule evaluator makes use of a global stack data structure. There are two important global variables here: *evalstore* and *evalptr.* Evalstore is an array of stacks, which hold only real numbers. Evalptr is an array of indexes into these stacks. A rule is evaluated selector by selector; the procedure *addmod* is used to evaluate a single selector and place its value on a stack. The other evaluator procedures are used to evaluate (combine) the numbers on a stack in a certain way. Procedure *max*, obviously, returns the maximum of all the numbers on a stack. Procedure *andresult* combines the numbers in a manner dependent on the global variable *g1and* (if the first rule group is being evaluated) or *g2and* (if the second rule group is being evaluated). The variables g1and and g2and are set in procedure init; the rule evaluator knows which rule group is being evaluated by the value of the variable *groupnum*, which is set by the rule modules when they are called. Procedure *orresult* works in a manner similar to *andresult.* All of the *ruleeval* procedures have a parameter which gives the stack number of the stack to be operated on (i.e. an index into the array *evalstore*). All the routines except addmod clear the stack they operate on.

The rule evaluation routines use no external files.

4.3 The Rule Units

The rule modules are produced by the ADVISE parser. There is one rule module for each rule group. Each module consists of one (interface) procedure; group one's procedure is called *firerule* and group two's is *fire2*. These procedures take one argument, the rule number to be fired. The procedures consist of a single, very large case statement, each rule being one case. Each case consists of a set of assignment statements and rule evaluator calls.

4.4 Unit PLANTHELP

This module consists of a single procedure which is simply a large case statement. There is one case for each descriptor. A case contains the help paragraph for the corressponding variable.

4.5 Unit INPUTOUTPUT

Each of the major PLANTIO procedures provides one of the options available from the PLANT/ds options page. These procedures are: *printh, writename, printvars, listh, listr, prule, helpcontrol, option_p* and *final_p.*

Procedure option-p is the main option-page routine (final_p controls the final option page and operates in a similar manner); it prints the option page on the screen, reads the user input and calls the appropriate procedure as described below. The only exceptions to this are options "c" (change answers on earlier question forms) and "q" (quit the consultation). In the case of option "c", option-p sets the call-by-reference parameter *back_up* to true and exits. In the case of option "q", the global variable *quit_all* is set to true and option-p exits.

Procedure printh is used to print out a set of hypothesis names. Its parameters are a set of rule numbers and a real number. The hypotheses are printed with their current confidences divided by the real number parameter.

Procedure writename is simply a case statement, one case for each disease (decision class). Each case prints the name of one disease.

Procedure printvars prints variable (descriptor) names. This procedure also consists of a single case statement.

Procedure listh lists the hypotheses that are currently considered working hypotheses. It uses procedure printh to print these hypotheses. The working hypotheses are those whose numbers appear in the global set *goodrules.*

Procedure listr lists the hypotheses that have been rejected. It also uses procedure printh and global set goodrules.

Procedure helpcontrol controls the system help facility. There is one subsidiary procedure for each page of help available in the PLANT/ds system.

Procedure prule is used to print a decision rule on the screen. This procedure first presents the user with a list of all the decision rules (procedure *options*), then

allows the user to select a rule (based on the letter printed next to the rule). Once the rule is selected, prule looks the rule up in a data file (below) and prints it on the screen.

The only files used by module PLANTIO are those files containing the text of the decision rules. There are two sets of data files; one set contains the expert rules (rule group 1), and are named "expert#.text" (where # is a number between 1 and 4). The other set of files contains rule group 2 and are named "mach#.text".

When prule wants a rule printed, it calls procedure *openr*, which decides what file the rule is in. It assigns this file to the pascal file variable *ruledata*. Procedure *findrule* is then called to print the rule. The rules are stored in text format in the data files. Rules are separated by lines containing the single character '#'. The first line of each data file contains numbers corresponding to the line numbers of the rules in the file. So, if the first line of the file is "2 15 30 45", then the first rule in the file starts at line 2, the second at line 15, and so on. Procedure *findrule* is passed the number of the rule to be printed in the file. It reads the line number of the appropriate rule, executes that number of readln calls, then reads and prints lines until it encounters the character '#'.

4.6 Unit TBLIO

The tblio unit is accessed through the interface procedure doscreen. This procedure controls the table driven user interface to PLANT/ds.

Unit tblio contains a large number of type declarations. The most important of these is the record *anslot*, declared as follows:

```
anslot = record
        x : columns; {columns is a subrange of integers 0..80}
        y : rows; {rows is a subrange 0..24}
        ans : answers; {answers is a subrange corresponding to ordinals of
                                type hypertype}
end;
```

Each anslot record corresponds to a single position on the screen that will be the answer to a question. The fields contain the x position, y position and answer number of that answer (answer number is taken as the ord of the actual value of the answer). The anslot records are used in an array:

```
scr = array [locations] of anslot;
```

This array is the set of all answers on the screen. The variable of this type is *screen*. So, to move the cursor to the third answer on the screen, the following statement is executed:

```
gotoxy(screen[3].x,screen[3].y);
```

Another frequently used array is:

```
qstart : array[1..maxques] of anslot;
```

where maxques is the maximum number of questions allowed on one screen. This array contains the record corresponding to the first answer to every question.

Procedure doscreen acts as follows:

Call procedure *setup* to initialize screen variables and arrays.

Call procedure *putupscreen* to put the questions on the screen. This procedure accesses files *screen#* and */screen#* (where # is an integer) to retrieve the text of the questions (see below).

Call procedure x_in_answers to fill in questions the user has already answered. This procedure accesses the variables' structures to determine if they have already been answered.

Go to the current screen position and wait for input.

On receipt of user input, call the appropriate procedures. Repeat this step until the user types a carriage return or all the questions are answered.

The tblio unit uses two types of files. The first type, the *screen#.text* files, contain question forms that have been set up beforehand. These files contain numeric data at the start, indicating the location of each of the answers. This data is in the following format:

<number of questions on this form>
<question number> <number of answers> <first answer x position>
<first answer y position> <first answer number> <second answer x position>
<second answer y position> <second answer number> ...
.
.
.
<question number> <number of answers> ...

This data is terminated with a line containing the single character '#'. Following the numeric data is the actual text of the question form.

The second file type, the */screen#.text* files, contain individual questions. Each question is preceded by numeric data in a format identical to that given above except for two things:

1) There is no line <number of questions on this form>
2) The y position is given as an offset from the first line of the question.

Questions are separated by single lines with the character '#'. The questions are stored in a number of different files. The routine *openf* determines which file will be accessed.

4.7 Unit CONTROL

This module is accessed through the interface procedure *schema*. Procedure schema contains the main algorithm for the PLANT/ds control scheme. Schema

acts as follows:

> Call the procedure *prepare_table* to set up the next question form. Prepare_table looks through the main list of descriptors; if it finds one that is the head of a question form (as declared in the slinks.text file), then a pointer to that variable is returned. Otherwise, prepare_table looks for variables in a manner dependent on the stage of the consultation. If there are more than five hypotheses in the set *goodrules*, then prepare_table finds variables that appear in the most rules (up to a maximum of four variables). If there are less than five hypotheses, prepare_table searches for variables that appear in the hypothesis with the highest confidence degree.
>
> Unit tblio is called to present these variables to the user. If tblio returns true to the parameter *proceed_ahead*, i.e. tblio did not receive the "o" command, then propagate the answers to the questions (fire the rules that the variables appear in). Otherwise, call procedure option-p in unit plantio.
>
> If procedure option-p returns the variable *go_back* equal to true, call procedure backup_control to return the user to an already answered question form.
>
> If proceed_ahead is true, return to the first step. Otherwise, return to the second step with the same set of variables.

Because of its size, the unit *control* is divided into two files, *plantctrl* and *ctrl2*. The file ctrl2 is included in plantctrl using a UCSD *include* statement.

The control scheme procedures do not access any data files directly.

4.8 Main Program

The main program is quite simple. It consists of calls to procedures *init* and *schema* (described above), and a simple loop to ask whether the user wants to start another session. The main program file also contains routines to *dispose* of all the domain descriptors (in order to prepare for a new consultation), and a routine to print the introductory screen.

References

Boulanger, A., 1983, "The Expert System PLANT/cd: A case study in applying the general purpose inference system ADVISE to predicting Black Cutworm damage to corn", M.S. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign.

Michalski, R.S., 1974, "Variable-Valued Logic: System VL1", *Proc. of the 1974 International Symposium on Multiple-Valued Logics*, IEEE Catalog Number 74CHO845-8C, pp. 323-346.

Michalski, R.S. and Baskin, A.B., 1983, "Integrating Multiple Knowledge Representations and Learning Capabilities in an Expert System: the ADVISE System", *Proc. 8th IJCAI*, pp. 256-258.

Michalski, R.S., Baskin, A.B., Boulanger, A.G. and Seyler, M.R., 1983, "A Technical Description of the ADVISE Meta-Expert System", Internal Report, Intelligent Systems Group, Department of Computer Science, University of Illinois at Urbana-Champaign.

Michalski, R.S. and Chilausky, R.L., 1980, "Learning by Being Told and Learning From Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis", *International Journal of Policy Analysis and Information Systems*, Vol. 4, No. 2, pp. 125-161.

Michalski, R.S., Davis, J.H., Bisht, V.S. and Sinclair, J.B., 1982, "PLANT/ds: An Expert Consulting system for the Diagnosis of Soybean Diseases", *Proc. 1st European Conf. on Artificial Intelligence*, pp. 133-138.

Rodewald, L.E., 1983, "BABY: An Expert System for interpretive reporting in a neo-natal intensive care unit", M.S. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign.