AGRIASSISTANT: A NEW GENERATION TOOL FOR AGRICULTURAL ADVISORY SYSTEMS

by

T. Fermanian
R. S. Michalski

In Expert Systems in the Developing Countries: Practice and Promise,
Ch. K. Mann and S. R. Ruth (Eds.), Westview Press Publication, 1992.

Reprinted from:

Expert Systems in Developing Countries

Practice and Promise

EDITED BY
Charles K. Mann
and Stephen R. Ruth

Chapter 5, AGASSISTANT: A New Generation Tool for Developing Agricultural Advisory Systems

Westview Press
BOULDER • SAN FRANCISCO • OXFORD

5

AGASSISTANT: A New Generation Tool for Developing Agricultural Advisory Systems

Thomas W. Fermanian and Ryszard S. Michalski

Introduction

AGASSISTANT is a new generation expert system builder for personal computers in the domain of agriculture. While it may be used to construct expert systems in any domain, its inference system was designed specifically to handle the uncertainty found in many agricultural domains. It is an extension of a conceptual predecessor, PLANT/ds, an earlier expert system for the IBM PC which was concerned with the diagnosis of soybean diseases common in Illinois. PLANT/ds was the first agricultural expert system that had the capability inductively to learn rules from examples, in addition to acquiring them directly from an expert (Michalski and Chilavsky, 1980b; Michalski et al. 1982). Unlike PLANT/ds, in which one interacts with the VAX minicomputer to build and refine the knowledge base, one need not leave the PC environment, either in creating an expert system, or in getting advice from a system that already exists. The work here is also based to a large extent on the ADVISE meta-Expert System (Michalski and Baskin 1983).

Among the many important features of AGASSISTANT are:

Multiple means of creating and refining knowledge. AGASSISTANT can receive rules directly or acquire them through inductive inference.

The authors wish to acknowledge J.C. Fech and J.E. Haley for their assistance in conducting the WEEDER evaluation study and to thank B. Katz and J. Kelly for their programming support.

Probabilistic inference is employed for handling uncertainty of data and rules. It is of great use in agriculture, where the vagaries of nature make identification or diagnosis a probabilistic matter.

Implemented on a personal computer. This allows wide dissemination of the program to farmers and others in need who are unlikely to have access to larger systems due to a specially designed human-computer interface.

Menu-driven screens. The novice user can quickly come up to speed in building expert systems due to a specially designed human-computer interface.

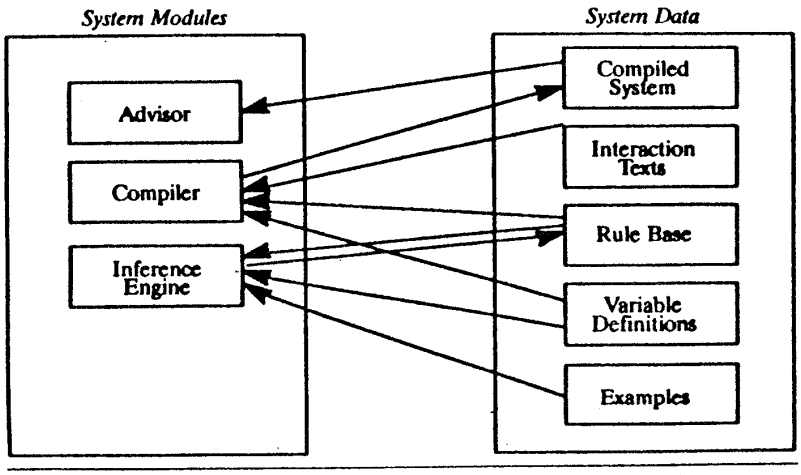
An Overview of the Program

The AGASSISTANT advisory system consists of a set of modules accessible through menu-driven screens (see Figure 5.1). The Advisor module takes as its input a compiled expert system, which it uses to create questions for the user, as well as to give advice on the basis of the answers to these questions. The Compiler module, in addition to parsing rules for correct syntax, creates a more compact version of the system for faster execution. Rules may be created by hand, or created through induction from examples. The Inference Engine module calls the program NEWGEM (Reinke 1984) to operate on examples, and variable definitions to produce rules. Additionally, this module may start with existing rules and improve them with examples (incremental learning), or optimize them according to differing criteria.

Knowledge Representation in AGASSISTANT

Knowledge is represented in AGASSISTANT primarily in the form of attribute-based logical calculus, called VL1 (Michalski 1975). Variables in

FIGURE 5.1. An Overview of AGASSISTANT



these rules may be nominal, linear, integer, or structured. The exact syntax and constraints on the rules are described in the section on knowledge acquisition. Here we will simply give the reader a feel for the way knowledge is represented so that he or she may better understand the following sections. A hypothetical rule that incorporates all of the allowed variable types appears in Figure 5.2.

The above rule contains two complexes (a conjunction of elementary conditions), the first of which consists of four conditions (also called selectors), while the second complex contains two selectors. Each selector is followed by a weight or confidence level (CL), which indicates the relative importance of the selector as a sole condition for making the decision. For example, if the only fact known is: "the crop shape is oblong," then the expert is 40% confident that the crop should be harvested. The method of combining these weights if more than one fact is known is explained in the Advisory module section. The action for this rule, namely that the crop should be harvested, depends on whether enough of the conditions are satisfied. The degree to which the set of conditions must be satisfied can be set as a system threshold. Unlike other inferencing mechanisms, this system can support a decision with only an approximate match of the evidence to the stated conditions.

The first condition will be satisfied if the shape of the crop is oblong. Since this is a structured variable, values of the variable are arranged in a hierarchy; thus this condition will be true if crop-shape takes the value of oblong, or any child of oblong. The second selector will be true if the temperature is between 65 and 75. The third selector will be fulfilled if soil-moisture is in the range of medium to high; there may be values in between medium and high (such as medium-high) which are implicitly included in this selector. The last condition will be satisfied only if the nominal variable sky receives the value sunny, out of a possible set of values including raining, cloudy, etc. The second complex of the rule, following the 'OR' consists of two selectors and says more or less that if you haven't brought in the crop by October, you should do so now if the weather is fair.

FIGURE 5.2 A Hypothetical Rule in the Knowledge Base of an Expert System

Crop should be harvested if:	*CL*
1. Crop-shape is oblong,	40
2. Temperature is 65 to 75,	45
3. Soil-moisture is medium to high	10
4. Sky is sunny.	5
OR	
1. Month is October,	75
2. Weather is fair.	25

A knowledge base for an expert system consists of a set of such rules. Rules may also be structured in a hierarchical fashion, i.e., a condition of one rule may be the action of another rule. The section on the exemplary expert system WEEDER outlines a small complete expert system.

Relevance to Agriculture

Many underlying principles of agricultural sciences can be experimentally measured in field experiments. The results of these experiments, however, show a normal abundance of variation in measured responses which is expected in nature. This natural variation has made the development of realistic models of agricultural systems difficult. Many assumptions are required for even the most simple crop or environmental model. Expert systems technology, for the first time, will offer a technique for working with fuzzy or uncertain knowledge.

Agricultural scientists often provide advice to agricultural managers on the basis of an evaluation of their incomplete knowledge and experience. This closely parallels the process of an expert system. Due to natural variability, agricultural knowledge bases are unstable and require continual modification to reflect current conditions or knowledge. Advisory systems which require the intervention of mainframe computers or centralized systems development cannot keep up with the rapid changes necessary. AGASSISTANT represents an expert system development environment that can be easily modified in the field. Therefore it can truly reflect any changes seen in the natural responses of the model in question.

While many agricultural production processes are inherently complex, a subgroup of production processes can be described adequately and converted to an appropriate knowledge base for use with AGASSISTANT. An example of these are pest or crop identification or pest damage diagnosis systems. In addition, simple designing of agricultural production systems would be an appropriate domain for AGASSISTANT. Often agricultural data are of a subjective, qualitative nature which might be more rapidly and thoroughly analyzed through symbolic processing techniques. AGASSISTANT represents a new technology in the form of a tool to assist agricultural scientists and managers to better interpret the observed phenomenon.

The Advisory System

While the Advisor module is most apparent to the user, at the heart of the AGASSISTANT is a flexible inference engine that runs on a compiled set of rules. The exact form of this file is described in detail in the first part of this section. The second part explains how individual rules are evaluated and how uncertainties are propagated through a hierarchical knowledge base. The third part explains the control structure of the system, i.e., the method by which the advisory system generates questions. This section explains the technical details upon which the advisory system is based.

System Representation

The rule parser takes a set of rules and a set of variable definitions and produces a file containing the compiled version of the expert system. This file consists of a cross-referenced version of the original system. Figure 5.3 shows a section of a rule base for an expert system, while Figure 5.4 illustrates the definitions of the variables involved.

The rules in the above system represent a section of a hypothetical expert system for crop management. Undoubtedly, an actual system, would contain many additional rules, (e.g. rules for fertilization, plowing, etc.) and each rule would be of greater complexity; this rule base is meant solely for illustrative purposes. The rules indicates that the crop should be harvested if the weather is good and the crop is ripe. Each of these conditions are in turn based on further conditions as indicated in the first two rules. The variables and their respective types and domains are shown in Figure 5.4. The compiled system appears in Figure 5.5.

Before explaining the desirability of converting the original system into its compiled form, the format of the file in Figure 5.5 will be explicated. The file consists first of a list of variables and information relevant to them. For example, the variable Crop-color is the fifth variable in the list (after Soilmoisture). Below the variable name is the relation used with this variable, and then the values associated with the variable obtained from the variable table. Following each value is a list of tuples, each containing four elements, *viz*, rule number, rule complex, selector within the complex, and a weight. For example, if Crop-color takes the value yellow, then selector 1 of complex 1 of rule 2 (rules are given numbers in order of appearance within the rule base) will be updated by 50% (the weights are represented as percentages with an additional digit of significance to reduce the possibility of roundoff error). Values may have more then one tuple following them if the variable value pair appears in more than one complex.

FIGURE 5.3 An Illustrative Rule Base

Crop should be harvested if:	*CL*
1. Crop is ripe,	60
2. Weather is good.	40
Crop is ripe if:	
1. Crop-color is yellow or green,	50
2. Crop-shape is oblong.	50
Weather is good if:	
1. Sky is sunny,	50
2. Soilmoisture is low,	50
3. Windstrength is very-mild to medium.	25

FIGURE 5.4 Variable Definitions for Illustrative System

Variable	*Crop*	*Weather*	*Soilmoisture*	*Crop-color*	*Windstrength*	*Crop-shape*	*Sky*
Type	*nominal*	*linear*	*linear*	*nominal*	*linear*	*structure*	*nominal*
Value 1	ripe	bad	low	yellow	very-mild	round	sunny
Value 2	unripe	fair	medium	green	mild	oval	cloudy
Value 3		good	high	brown	medium	pear	rainy
Value 4				black	mediumstrong	oblong	
Value 5					strong	short	
Value 6						long	
Value 7							
Value 8							

Following the variables, appear two lines of #'s, after which appear the rule actions. Each action has three lines of information. The first line is simply the action verbatim, as it appears in the rule base. The second line, possibly blank, contains the name of a text file (e.g., harvest.txt for the rule "Crop should be harvested") invoked if the rule associated with the action receives the highest confirmation at the completion of the advisory session. The third line consists of a number that indicates which variable, if any, is identical with the action, followed by a list of four-tuples, with the same ordering as described above, which list the locations of the action as conditions in other rules. For example, this line for the action "Crop-shape is oblong" begins with an 8 to indicate that this action variable, *viz.*, "crop-shape", is also the eighth variable in the variable list at the beginning of the file. Additional information indicates that the action also appears in rule 1, in the first selector of the first complex.

In the fourth and last line the order in which variables should be asked is presented. Following this is a list of arithmetic expressions found throughout the rule base, with the appropriate tuple list trailing each one. Finally, variables and questions are listed that will be asked during the advisory session when the system wishes to know the value of a variable.

One may contend that the information contained in the compiled system, with a few minor additions, is just a rehashing of the system in its original form. While this is correct, there is an important reason for representing the system in such a way. Consider what needs to be done to update a rule after a new value is associated with the variable. Suppose the system consists of n rules each with an average of s selectors. One must then search the entire rule space for the appearance of the variable-value pair, or perform ns searches. Additionally, in the worst possible case, one must search for the actions appearing

as selectors in other rules resulting in sn^2 matches, or ns matches for each of n rules. While both of these figures are polynomial quantities, they will nevertheless prove prohibitively large for a PC system if n is large. Thus the appearance of all values, as well as the location of all actions as conditions in other rules, are cross-referenced beforehand in the compiled file, resulting in almost no search. The exact method of updating rule confidence levels, once this information is provided, is examined in the following section.

FIGURE 5.5 An Example of a Compiled Expert System

Variables

```
##############
crop
is
ripe                111600
##############
Weather
is
bad
fair
good                112399
##############
Soilmoisture
is
low                 312250
medium
high
##############
Crop-color
is
yellow              211500
green               211500
brown
black
##############
Windstrength
is
very-mild           313250
mild                313250
medium              313250
strong
##############
Sky
is
cloudy
rainy
sunny               311500
##############
Crop
is
ripe                212600
##############
```

Rule Actions

```
##############
##############
Crop should-be harvested
harvest.txt

38

Crop is ripe

8111
125

Weather is good

3112
467
##############
12345678
##############
```

$$ Arithmetic expressions

```
$$ (Crop-width * Crop-length) > 12 212500

##############
```

Variables and associated questions

Weather	What is the weather like?
Soilmoisture	To what extent is the soil saturated?
Crop-color	What color is the crop?
Windstrength	How strong is the wind?
Sky	What is the appearance of the sky?
Crop-shape	Is the shape of the crop oblong?
Crop-length	What is the length of the crop?
Crop-width	What is the width of the crop?

FIGURE 5.6 General Rule Format

Action is xxxx if:	*confidence level*	
1. variable1 is value1,	60	Complex1
2. variable2 is value2 or value3,	40	
3. variable3 is value4 to value6.	20	
OR		
1. variable4 is value7,	50	Complex2
2. variable5 is value8.	50	

Inference Mechanism

Rules have the general format shown in Figure 5.6. Since the conditions in rules are annotated by weights of confidence levels to represent strength of evidence in favor of the decision, it is not sufficient simply to invoke the standard laws of deductive inference in evaluating confidence levels of the rules. The next section describes the method for evaluating individual rules, while the one following it shows how rules are evaluated in a hierarchical system.

Individual Rule Evaluation. The complex of a given rule is easy to evaluate given the form of the compiled system. A sum is maintained for each complex of each rule, and this sum is augmented by the amount indicated by the tuple associated with the satisfied variable-value pair. The general formula for evaluating a complex is:

$$(1) \qquad \frac{\Sigma \text{ (weights of satisfied selectors)}}{\Sigma \text{ (all weights in the complex)}}$$

For example in complex1 of the general rule in Figure 5.6., if variable1 had the value of value1, and variable 2 took the value value3, while variable3 the value value9 (not in the linear range value4) then complex one of this rule would have the value of:

$$(2) \qquad \frac{60+40}{60+40+20}$$

or 83%. Selectors with internal disjunction (selector with a set of disjunctive values) are assumed satisfied if any of their values is the current value of the variable. Selectors with linear variables are satisfied if the variable takes on any value within that range inclusive of the end points. Finally, selectors with structured variables are true if the variable receives the indicated value or some child of that value. For example, the condition "Crop-shape is oblong" will be true if either oblong, short, or long is selected.

Evaluation of rules with multiple complexes is performed by recursively taking the probalistic sum (referred to as "psum" evaluation in ADVISE [Michalski and Baskin 1983]) of the *n*th complex with the psum of the first n-1 complexes. The formula for this is:

$$(3)\quad psum(V(n),V(n-1),\ldots,V(1)) = V(n) + psum(V(n-1),\ldots,V(1)) - V(n)*psum(V(n-1),\ldots,V(1)),$$

where $V(x)$ is the evaluated value of complex x.

Other evaluation schemes such as taking the complex with the highest value are possible but are not included in this version of the system. The virtue of the psum scheme is that it fits well to the intuitive notion that if any complex is completely satisfied, the rule also will be, a notion that the probability of two independent events occurring is the probalistic sum of the event probabilities.

A rule is considered to be satisfied if its confidence level goes above a threshold, experimentally set at 85%. Notice that if all selectors are assumed to have equal weights and the threshold is set at 100%, then the evaluation schemes described here collapse into formal logic expression.

Evaluation of Rules in a Hierarchical Rule Base. Selectors in rules which are in turn actions of another rule receive the value of the rule times the weight for that selector. Thus in Figure 5.3 if one assumes that the sky is sunny in the third rule, and that the confidence that the weather is good is 50%, the confidence that the weather is good in the first rule is 50%*40%, or 20%.

A hierarchical rule base contains a partial ordering among the rules. The rule evaluation module topologically sorts the rules according to this partial ordering before the advisory session begins, and uses the resulting order to evaluate all rules after each new value is entered. This ensures that all weights are propagated in the correct sequence. Notice, that if one assumes that all selectors of all rules receive equal weights, and if the threshold of rule firing is 100%, the inference scheme emulates a standard forward-chaining inference engine.

Control Mechanism

The Advisor module of AGASSISTANT has two control mechanisms, that is, schemes which determine the order in which questions are asked. The first method applies to rule bases which are flat, or non-hierarchical in nature, and is known as the utility scheme. The second mechanism, for hiearchical rule base, is a backtracking scheme.

Utility Control Scheme. The utility control scheme is represented in Figure 5.7. At the start of the advisory session questions are asked for variables in order of the utility of the variables. The utility measure, precalculated and found in the compiled system file, reflects the degree to which the variable will affect the confirmation level of all rules. Those variables appearing in the most places in the rules are given the highest utility. The advisory session continues

until the confirmation level of any rule goes above a lower threshold, experimentally set at 15% 1-upper threshold. When this occurs, the system will focus on that rule by asking for the values of all variables relevant to the rule. This continues until the rule is rejected, or all variables for that rule are exhausted. At this point the system will focus on another rule which is above the upper threshold, or if none exists it will return to the utility measure. The entire process continues until all rules are rejected or confirmed. This may require that all variables be queried for, but usually occurs much sooner. In general, a rule is confirmed if it has a confidence level above the confirmation threshold, while it is rejected if it cannot possibly be confirmed, no matter what the values for the remaining variables are. The system makes use of a method of keeping a "negative" confidence for all rules; thus there is no need to determine dynamically whether a rule is rejected after each new answer.

Backtracking Control Scheme. The backtracking control scheme is automatically invoked if the rule base is hiearchical. It is represented in Figure 5.8. The system begins by asking questions for variables with highest utility and continues in this fashion, until the user answers "don't know" for a variable

FIGURE 5.7 Utility Control Scheme

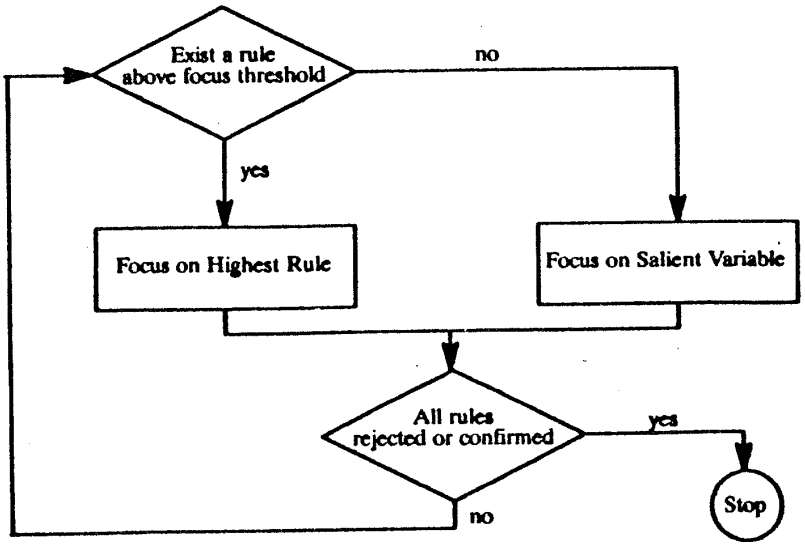
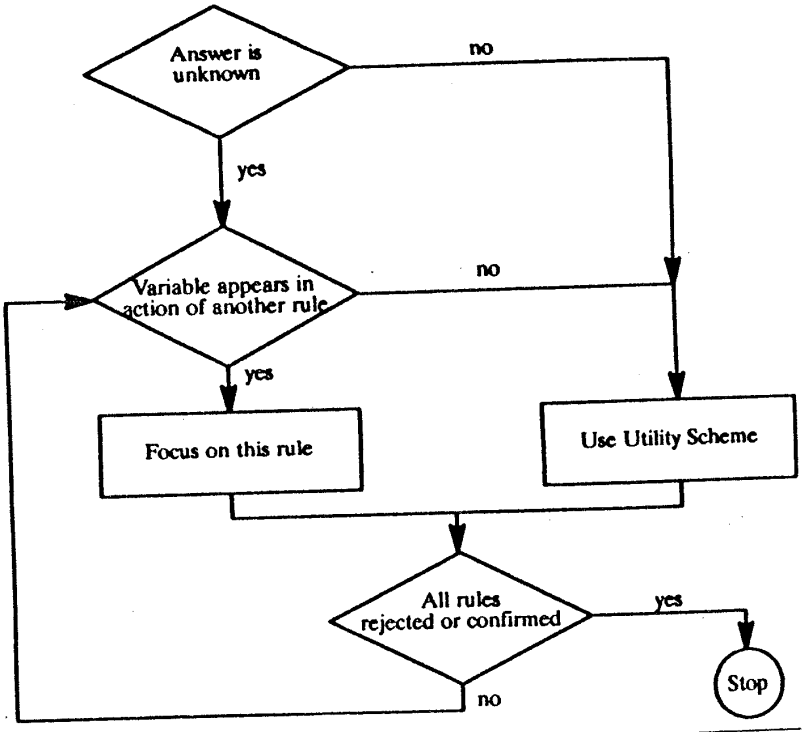


FIGURE 5.8 Backtracking Control Scheme

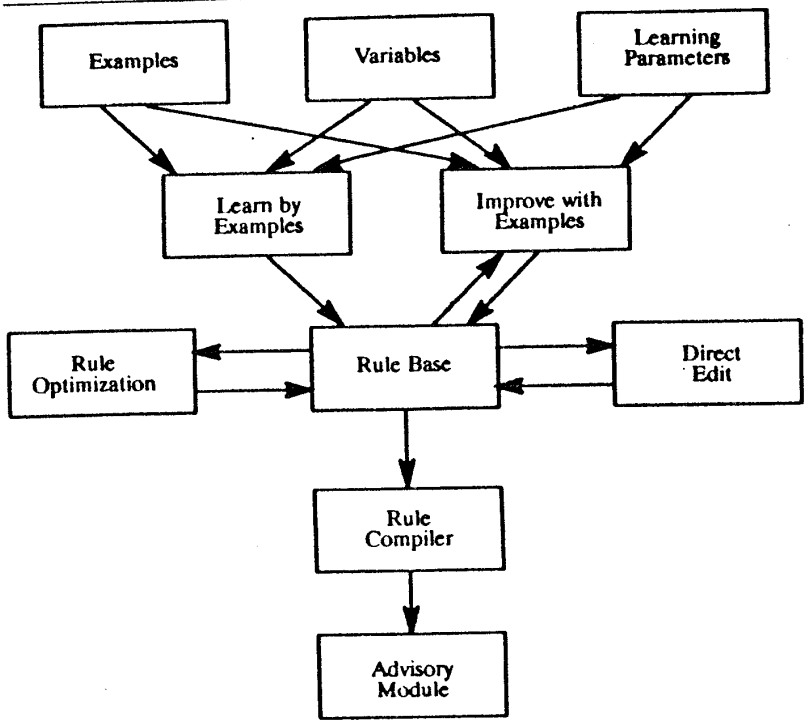


which also appears in the action of some rule in the system. For example, if the user answers "don't know" for the variable Weather in the rule base of Figure 5.3, then the system will attempt to infer the value of Weather by asking for the values of Sky, Soilmoisture, and Windstrength. The process continues recursively until the system is able to find the value for the original variable it was focusing on, in this case Weather, at which point it returns to the utility scheme. The system will continue to query the user until all rules are either rejected or confirmed.

Knowledge Acquisition Facilities

Figure 5.9 illustrates the knowledge acquisition facilities of AGASSISTANT. As stated at the outset, knowledge is represented in the system in the form of rules in the VL1 syntax. These rules can be acquired in four possible ways. They

FIGURE 5.9 Knowledge Acquisition in AGASSISTANT



can be learned from examples, improved with examples, optimized, or edited directly. It is also possible to use these methods in combination. For example, one common procedure is to edit rules directly, and then optimize them to remove superfluous information. Once acquired, rules are compiled and serve as input to the advisor module: in effect the system's interface with the user. The various facets of knowledge acquisition are described below.

Direct Editing of Rules

One way to enter knowledge into the system is to handcraft it to the system. To use this method an expert must be able to express knowledge in the form of rules. This notoriously difficult problem is often referred to as the knowledge acquisition bottleneck (Michalski and Chilausky 1980b). For example, one may be a perfectly adequate driver and yet have difficulty expressing

this knowledge in rule format. Nethertheless, there are many domains in which direct entry of rules is appropriate. In AGASSISTANT, this is accomplished in two steps. First, the relevant variables are entered into the variable editor. The variable type, and its domain, are specified within the editor. Then, upon selection of the direct edit function within AGASSISTANT's menu system, a child process will be created. This process consists of the default editor (whose name is stored in the file 'AGEDITOR.TXT') with the file to edit being concatenation of the current system name and the extension .RLE. The command processor of MSDOS will return the user to the AGASSISTANT system upon termination of the edit. The user then typically will attempt to compile the newly created rule base, and will repeat the edit-compile cycle if errors are indicated by the compiler.

It is important to note that the only way that *AGASSISTANT* currently can acquire a hierarchically structured rule base is by directly creating it. This is because the current learning program used in the system is not capable of constructing a knowledge base with intermediate layers of knowledge. One can work around this, by learning a set of subconcepts, and then learning higher-order concepts or directly entering such concepts. For example, one could first learn a set of rules describing diseases that afflict a given species. One could then do another experiment to learn the best treatment for the plant with one of the variables in this experiment being the disease, if any, of the plant. One could then concatenate the two resulting rule sets within the editor to produce a structured knowledge base. This method can be used to produce a rule group of arbitrary depth, although it is likely that any complex domain will consist of a mixture of expert and induced rules.

Rule Learning

The underlying algorithm for all the learning facilities within AGASSISTANT's Inference Engine module is the NEWGEM program (Reinke 1984). At the heart of NEWGEM is the *Aq* quasioptimimal covering procedure. As *Aq* is described in detail in many previous papers, (see e.g. Michalski 1973) we will not go into great detail about it here. Suffice it to say that *Aq* works by attempting to find a rule which covers all of the positive events and none of the negative events, positive events being those belonging to the decision class under consideration, and negative events being all others. It does this by selecting a seed event within the set of positive events and extending it against successive negative events until it covers none of the negative events. Extending a partial cover against a negative event simply means specializing it so that it no longer covers the negative event if indeed it did to begin with. This process is continued until a cover or disjunction of covers is produced for all of the original positive events.

Learning Rules from Examples. Learning from examples is one of the most well-explored areas in machine learning (Dietterich and Michalski 1983; Michalski 1986). In this form of learning, a teacher provides characteristic examples and their respective decision classes to the learner. The task is then to create a set of rules which classify the given events. While simple in principle, this method of building descriptions of concepts is often powerful in practice. For example, in a now famous case it was shown that inductively derived rules for soybean disease diagnosis outperformed expert given rules (Michalski and Chilausky 1980a).

AGASSISTANT combines variable definitions as found in the variable table, events as found in the data table, and parameters as set in the parameters table to form the file SPECIAL.GEM which is then passed to the NEWGEM learning program. The parameters for the program determine various aspects of the rule creation process, including the breadth of the beam search used by the *Aq* algorithm, the lexicographic functional which is used in sorting candidate hypotheses, and the extent to which produced rules will be trimmed. (See Reinke (1984) for more information on the meaning of the NEWGEM parameters.) The NEWGEM module takes the input from SPECIAL.GEM and sends its output to the file LAST.GEM, which is then copied to the file {SYSTEM NAME}.RLE if the user decides to save the produced rules.

Each selector in a produced rule is associated with a weight. These weights are calculated by the following formula, and then normalized so that that the weights of a given complex sum to 100.

$$weight = \frac{pe}{pe + ne} \tag{4}$$

where *pe* is the number of positive events covered, and *ne* is the number of negative events covered by the selector. Thus, the weight produced represents the probability that the given decision class is indicated given that the selector is satisfied.

Improving Rules with Examples. AGASSISTANT is capable of improving its knowledge as new examples are presented to it. This method is known as incremental learning with perfect memory, and the algorithm for performing this task is presented in detail in Reinke (1984). Summarizing his description, we find the method to be a straightforward extension of the *Aq* algorithm. First, the existing cover for a class of events is specialized to take into account new negative examples. This new cover is then used as the original seed event for *Aq*.

As can be seen in Figure 5.9, the input for rule improvement is constructed from the existing rule base, the table of examples (which includes the newly entered examples), and the variable table. This file is then sent to the NEWGEM learning program which detects the presence of input hypotheses and therefore runs *Aq* incrementally. As Reinke mentions, one must be careful when using this learning mode, since it is likely that the com-

plexity of the output rules will increase, although the amount of this increase depends on the nature of the new examples. Clearly, if the new examples are for the most part in existing decision classes, the rules will not change that drastically assuming that the original induction was performed on a statistically large enough sample of events. If the new events fall into new classes, or if the original events came from a small subsection of the true problem space, then the rules will exhibit a proportional increase in complexity. The chief advantage of the incremental learning method presented here is the speed increase of the induction process. Incremental learning was only partially implemented in AGASSISTANT.

Rule Optimization. The system allows a user to optimize selected rules according to certain criteria. One form of this is a conversion of characteristic rules to discriminant rules. Michalski defines characteristic rules as those that specify common properties of the members of a class, and discriminate rules as those which have only enough information to distinguish one class from another or another set of classes (Michalski 1983). Here characteristic rules will refer to any that are not discriminant. For example, expert-created rules typically fall somewhere between the discriminant and characteristic categorizations. Indeed, one common use for this facility is to compress rules provided by an expert to their discriminant versions.

The method for rule optimization takes advantage of facilities already provided by the NEWGEM and submitted as input hypotheses with no corresponding input examples. If the rule type parameter is set to produce discriminant rules, these rules will be generalized to discriminant form.

Figure 5.10 shows the results of converting three characteristic rules to discriminant form. Notice that the optimized rules contain only the information necessary to distinguish between the three actions, in this case, the values of the variables shape and texture. This method of rule optimization only makes sense in the context of an expert system if one is confident that the values of the discriminatory variables will be known by the user of the expert system. If this is not the case, the system will perform better if the rules are left in their characteristic form.

FIGURE 5.10 Rules Before and After Optimization

Action is one if:	Action is two if:	Action is three if:
Before Optimization		
1. Color is red, 2. Size is small, 3. Shape is round, 4. Texture is smooth.	1. Color is red, 2. Size is small, 3. Shape is square, 4. Texture is rough.	1. Color is red, 2. Size is small, 3. Shape is square, 4. Texture is smooth.
After Optimization		
1. Shape is round, 2. Texture is smooth.	1. Shape is square, 2. Texture is rough.	1. Shape is square, 2. Texture is smooth.

Rule Compilation

Once acquired, rules must be compiled if they are to be used in an advisory capacity. The compiler has two chores. One is to check the syntax of the rules and provide the user with the appropriate error messages if the rules do not parse. If the rules parse successfully, the compiler will then create a file suitable for the Advisor module to ask questions and give advice. The exact nature of this file was detailed earlier and will not be repeated here.

Figure 5.11 below contains the complete grammar for rules in the system. Summarizing this figure, a rule consists of a condition part and an action part. A condition consists of the disjunction of a set of complexes, which in turn consist of the conjunction of selectors. A selector consists of a variable, a relation, and either a value, a disjunction of values, or a range of values, plus an optional weight.

The rules are parsed in a straightforward way, that is, the parser is structured as a finite-state machine, with each of the elements of the machine optionally being another sub-machine.

FIGURE 5.11 Grammar for Rules

< rule >	:=	< action > < condition >
< action >	:=	< variable > < relation > < value > "if:"
< condition >	:=	< complex > "OR" < condition >
	:	< complex >
< complex >	:=	< selector > < complex >
	:	< selector >
< selector >	:=	< variable > < relation > < value-list > < terminator >
	:	< a-expression > < a-relation > < a-expression > < terminator >
< variable >	:=	string of letters
< relation >	:=	string of letters
< value-list >	:=	< value > "or" < value-list >
	:	< value > "to" < value >
	:	string of letters
< terminator >	:=	< termchar >
	:=	< termchar > < weight >
< termchar >	:=	" "
	:	"."
< weight >	:=	natural number
< r-number >	:=	real number
< a-relation >	:=	"<," "< =," "=," ">," "> ="

An Exemplary System: WEEDER

To illustrate the use of AGASSISTANT, we describe its use to develop an advisory system WEEDER. In the design of an effective weed-control program for managing turf it is first necessary correctly to identify the species of weed(s) present and to determine the extent of their population. Morse (1971) outlines five basic identification methods for determining unknown plant species: (i) Expert determination, which is generally regarded as the most reliable of all identification techniques. This method merely transfers the responsibility of identification to an appropriate expert. This service can be

slow and costly, and is often limited by the availability of an expert. (ii) Immediate recognition, approaches expert determination and accuracy. This is the ability of an individual to recognize an unknown weed by past examples of identification. For some taxonomic groups and immature plants, however, this method of identification is very difficult and in all cases requires extensive past experience. (iii) A comparison of an unknown specimen with identified species or illustrations. It offers a rapid, simple diagnosis and is often useful for many weeds commonly found in native populations. (iv) An identification key which is based on the development of appropriate descriptive phrases of morphological or biochemical characteristics. Identification keys generally take the form of groupings of similar characters from which the user must select the character which best matches that present on the unknown sample. The selection of this character then leads to the next set of identifying characteristics. This process is followed until enough characteristics have been identified to suggest the identification of the specimen. (v) The last identification technique is a diagnostic table or polyclave. Diagnostic tables are a matrix of rows of species and columns of identifying characteristics. Users of a diagnostic table can identify the listed characteristics in any order they wish.

Morse (1971) lists two major faults of identification keys: (i) They require a user to utilize certain characteristics whether or not they are convenient or can be identified; (ii) They implicitly rely on rigid descriptions of specimens. Occasional variation in a population can cause gross misidentification.

The use of expert systems techniques offers a new, unique method for assisting with species identification. The relative merits of an expert or advisory systems is the ability to select answers or queries about characters that are available on the unknown specimen. They can operate on various levels of uncertainty providing a more efficient mechanism for identification, particularly for immature plants which are even difficult for experts to identify.

Development of WEEDER

In order to prepare the knowledge for use with AGASSISTANT, a matrix was developed including each potential grass weed. Eleven identifying characteristics, both vegetative and floral were determined for each weed. The information was obtained from many sources: textbooks, weed identification manuals, botanical manuals, and the authors' experience. Variable names, types, and sets of values is shown in Figure 5.12. The characteristics selected were those thought to be most easily recognized in the field without supportive equipment. (Shurtleff et al. 1987) Figure 5.13. presents a typical rule. This rule consists of 10 selectors. Each selector is associated with a weight or confidence level (*CL*) that indicates the relevant importance of the condition in making the decision. Notice that these weights need not add up to 100; they are normalized by the system. The weights give a rough estimate of the

importance of each of the conditions in discriminating between the rules. These weights were then refined by the domain expert (T. Fermanian).

Rules for WEEDER were developed utilizing the NEWGEM module of *AGASSISTANT*. Separate rule sets were formed, first by inducing a set of characteristic rules, m and then by inducing a set of discriminant rules. The most appropriate rules from both sets were then modified, utilizing expert experience and written to a single rule set used in the initial evaluation.

Grass weed identification in turf is generally only available through the use of vegetative characteristics. This is due to the frequent mowing of the turf which often removes any floral portions of the plant. WEEDER allows the user to select either vegetative or a combination of floral and vegative characteristics at the beginning of each session. This is done through a "does-not-apply" question which is always asked first in the consultation.

"Does not apply" questions were established in order to provide a meaningful subset of variables for WEEDER to act on. The question "Are seedheads or flowers present?" to which the user responds "yes," "no," or "don't know" begins each session. If a "don't know" answer is given, then all identifying characteristics are asked. If "yes" is answered, then eleven of the possible characteristics are presented to the user. If "no" is answered, which is

FIGURE 5.12 Variable Names, Types, and Values for WEEDER

VARIABLE	*Vernation*	*Auricle*	*Ligule*	*Sheath*	*Collar*	*Blade-width*
TYPE	*nominal*	*nominal*	*nominal*	*nominal*	*nominal*	*linear*
1	folded	absent	ciliate	compressed	narrow	fine
2	rolled	short	round	round	divided	medium
3		claw-like	truncate	closed	broad	coarse
4			acute			
5			toothed			
6			acuminate			
7			none			
8						

	Habit	*Glumes*	*Disart*	*Awns*	*Florets*	*Flower*	*Nerves*
	nominal	*nominal*	*nominal*	*nominal*	*integer*	*nominal*	*integer*
	bunch	shorter	above	absent	1	panicle	1
	rhizome	longer	below	present	3	raceme	3
	rhiz-stolon			bifid	5	spike	5
	stolon						

FIGURE 5.13 A Rule for Identifying Stinkgrass

Weed is Stinkgrass if:	*CL*
1. Florets are 10 to 12,	85
2. Flower is panicle,	70
3. Collar is narrow,	60
4. Blade-width is medium,	55
5. Habit is bunch,	50
6. Sheath is compressed,	50
7. Vernation is rolled,	35
8. Glumes are shorter,	30
9. Florets is 1,	30
10 Disart is below.	25

FIGURE 5.14 Does Not Apply Conditions in WEEDER

If seedheads are present then
    Auricle, and Blade-width do not apply.
If seedheads are not present then
    Florets, Flower, Awns, Disart, and Glumes do not apply.

the usual situation for turf, then seven characteristics are presented—only those pertaining to vegetative portions of the plant. Paraphrased versions of these do not apply conditions are shown in Figure 5.14.

WEEDER Evaluation

In order to evaluate the absolute efficiency of WEEDER in drawing the correct conclusion, it was necessary to develop a program to determine the minimum number of variables required to identify each species. This program determined which groups of variables would provide a CL of threshold or greater value for a chosen rule. This program was run external to AGASSISTANT and was used to determine the maximum set of variable combinations for each rule in WEEDER.

Gower and Barrett (1971) state that the most efficient determination of an unknown species using an identification key is to use identifying variables which divide potential species into equal binary groups. They therefore suggested an equation to represent the minimum theoretical number of decisions necessary when using a dichotomous identification key.

$$\textit{Minimum number of decisions} = \log_2 n$$

Where n represents the total number of species considered. If a dichotomous key was constructed to identify the 37 grasses of WEEDER, a minimum of five variable decisions would have to be made for a positive identification. This minimal number of decisions would therefore provide the most efficient use of

the key. This would require each decision to equally divide the species which is not possible for a key using the chosen variable for the 37 species in WEEDER. In practice most identification keys do not provide this optimum efficiency (Pankhurst 1978).

Only the subset of variables describing vegetative characters was used in the evaluation of WEEDER. For the 37 grasses, there was a maximum of 16 sets of variables providing for the correct identification of any one grass. For four species only a single set of variables provided its identification. There were a total of 145 different sets of variables which represented correct identification. There were a total of 145 different sets of variables which represented correct identifications of any species with a mean of four sets for each species. The average number of variables necessary to correctly identify each grass species and its accompanying mean CL is listed in Figure 5.15. An identification was made when the CL was 85 or greater. The mean CL for all identifications was 92.

With a mean number of five variables required for each identification, WEEDER's efficiency was similar to the theoretical maximum efficiency for a dichotomous key to identify the same species. A dichotomous key with this level of efficiency has not been constructed using the same set of variables and values for the species in WEEDER. The maximum number of variables required for a correct identification of any species was seven. Over one-half (59%) of the identifications required five or fewer variables. For most species (96%), the recognition of a maximum of six variables was required for its identification. Since dichotomous keys generally do not perform at the theoretical maximum efficiency for identifying species, WEEDER shows excellent potential as a grass identification tool.

FIGURE 5.15 Mean Number of Variables and Average CL Required to Identify Any of 37 Grass Species Using WEEDER.†

	No. of variables‡	*CL*
Mean	5§	92
Minimum	4	88
Maximum	6	99
SE¶	0.1	0.5
CV (%)#	11	3

† For each unknown specimen an identification was considered correct when the CL was ≥ 85.
‡ Mean number of variables necessary for the identification of each species.
§ Mean of 37 species means.
¶ Standard error of mean.
# Coefficient of variation.

The WEEDER advisory system provided an excellent exemplary system and helped to test some of the proposed capabilities of AGASSISTANT. Specifically, WEEDER was able to provide a means for the satisfactory identification of 37 grass species commonly found in turf through a minimal number of decisions. In most cases, multiple sets of variables could be used to identify a single species.

Validation of WEEDER

In order to measure the relative efficiency of WEEDER in identifying unknown grasses, a study was conducted in which individuals were asked to identify four unknown grasses. Four grasses were selected randomly from a set of fifteen grass species commonly found in central Illinois. The four species selected were: creeping bentgrass (*Agrostis palustris* L.), perennial ryegrass (*Lolium perrenae* L.), zoysiagrass (*Zoysia japonica* L.) and large crabgrass (*Digitaria sanguinalis* (L.) Scop.).

Forty-one volunteers were assigned to one of two groups. If they had previous experience in plant diagnosis or formal training in plant science, they were separated from those volunteers who had no biological or plant science training or experience. Each individual randomly selected two of the four unknown weeds for identification using WEEDER, the other two weeds were identified using a diagnostic key, a commonly used tool (Shurtleff et al. 1987).

Along with the four unknown grass samples, each participant was supplied with a low-power dissecting microscope, appropriate probes and dissecting equipment, and a book with representative diagrams of all the potential configurations of morphological characters. Each individual was allowed up to 30 minutes per weed for identification. Fifteen minutes was reserved for a demonstration of each character and an explanation of how it could be identified. For the plants identified through the diagnostic key, each participant only supplied their first and possibly a second choice, as suggested by the key. Grasses identified with WEEDER, however, offered participants the ability to indicate the configuration chosen for each plant character. A frequency analysis of the correctly identified grasses was then conducted to determine their fit to the χ^2 distribution.

WEEDER has the ability to rank all the grasses in its knowledge base from the species most likely to represent the unknown grass to the one least likely. Figure 5.16 presents the mean percentage of identified grasses across all species using either identification tool (WEEDER or the identification key). The identification key, a tool commonly used by the participants in the study with plant science training, showed the highest average rate of success (20%) for identifying a species in the initial evaluation. The mean success rate of participants with plant science training in identifying any species using

FIGURE 5.16 Mean Percentage of All Correctly Identified Grass Species Using Either WEEDER or an Identification Key by Participants with Either Plant Science Training or Without Plant Science Training.

Selected Frequency Group	Mean correctly identified species	
	Initial rules	Modified rules
Identification tool		
WEEDER†	11	50
Identification key	20	20‡
χ^2	2.3	16.8
	NS	••
Participant group		
Plant science training §	19	39
No plant science training	13	32
χ^2	2.7	1.9
	NS	NS

•,•• Significant at the 0.05 and 0.01 levels, respectively. NS = not significant at the 0.05 level.

† For both participant groups.

‡ Since the modification of an identification key is not practical the same values were used for the "Modified rules" evaluation.

§ For both identification tools.

WEEDER was 15% and 7% for participants without plant science training using WEEDER (not shown in Figure 5.16.) After this initial experiment it would appear that the test group was not successful using WEEDER to identify the selected species. A closer evaluation of the answers selected by those using WEEDER showed a consistent problem in correctly determining the value of a few morphological characters. The natural variation in the growth of the selected species and their juvenile state made the identification of fine characters, such as the ligule, very difficult.

Rules for identifying the four grass species examined were modified (Figure 5.17) through the incremental learning facility in AGASSISTANT. The test groups answers were used as examples to improve the original rules. The results of these changes showed a very large gain in the percentage of correctly identified grasses. On the average for both groups, the percentage of correctly identified grasses rose from 11 to 50% when using WEEDER, as compared to the 20% mean for grasses correctly identified with the identification key (Figure 5.16.)

While no significant indication of dependence on either the identification tool or participant group was shown (Figure 5.16) using the initial rules, a very significant dependence ($P < .01$) on the identification tool used after rule modification indicates the potential advantage of WEEDER over the identification key for all participants. No significant indication of dependence on

Figure 5.17 Rules Representing Four Grass Species Used in the Evaluation of WEEDER Before and After Their Modification

Initial			Modified†		
Weed is Bentgrass if:		cf‡	**Weed is Bentgrass if:**		cf
1.	Ligule is round,	65	1.	**Ligule is round or toothed,**§	65
2.	Sheath is round,	65	2.	**Sheath is round,**	65
3.	Glumes are longer,	65	3.	**Glumes are longer,**	65
4.	Habit is stolon,	60	4.	**Habit is stolon,**	**40**
5.	Disarticu is above,	55	5.	**Disarticu is above,**	55
6.	Collar is narrow,	50	6.	**Collar is narrow,**	**70**
7.	Florets is 1,	45	7.	**Florets is 1,**	45
8.	Flower is panicle,	45	8.	**Flower is panicle,**	45
9.	Blade-width is fine,	35	9.	**Blade-width is fine,**	**70**
10.	Vernation is rolled.	30	10.	**Vernation is rolled.**	**70**
Weed is Per-ryegrass if:		cf	**Weed is Per-ryegrass if:**		cf
1.	Ligule is round,	85	1.	**Ligule is round or truncate,**	**25**
2.	Auricle is short,	80	2.	**Auricle is short,**	80
3.	Florets is 6 to 10,	80	3.	**Florets is 6 to 10,**	80
4.	Flower is spike,	75	4.	**Flower is spike,**	75
5.	Vernation is folded,	50	5.	**Vernation is folded,**	50
6.	Habit is bunch,	40	6.	**Habit is bunch,**	40
7.	Collar is broad or divided,	35	7.	**Collar is broad or divided,**	**75**
8.	Sheath is compressed,	30	8.	**Sheath is compressed,**	**70**
9.	Blade-width is fine to medium,	30	9.	**Blade-width is fine to medium,**	**70**
10.	Disarticu is above,	35	10.	**Disarticu is above,**	35
11.	Glumes are shorter.	25	11.	**Glumes are shorter.**	25
Weed is Zoysiagrass if:		cf	**Weed is Zoysiagrass if:**		cf
1.	Habit is rhiz-stolon,	80	1.	**Habit is rhiz-stolon or rhizome,**	80
2.	Glumes are longer,	80	2.	**Glumes are longer,**	80
3.	Awns are present,	75	3.	**Awns are present,**	75
4.	Flower is spike,	70	4.	**Flower is spike,**	70
5.	Sheath is round,	70	5.	**Sheath is round,**	70
6.	Ligule is ciliate,	60	6.	**Ligule is ciliate,**	60
7.	Florets is 1,	55	7.	**Florets is 1,**	55
8.	Blade-width is medium,	50	8.	**Blade-width is fine to medium,**	50
9.	Collar is broad,	50	9.	**Collar is broad,**	**70**
10.	Disarticu is below,	45	10.	**Disarticu is below,**	45
11.	Vernation is rolled.	35	11.	**Vernation is rolled.**	**75**
Weed is Lg-Crabgrass if:		cf	**Weed is Lg-Crabgrass if:**		cf
1.	Ligule is toothed or acute,	65	1.	**Ligule is toothed or acute,**	65
2.	Blade-width is course,	60	2.	**Blade-width is medium,**	**60**
3.	Flower is spike,	60	3.	**Flower is spike,**	60
4.	Sheath is compressed,	50	4.	**Sheath is compressed,**	50
5.	Habit is bunch,	40	5.	**Habit is bunch,**	**60**
6.	Disarticu is below,	40	6.	**Disarticu is below,**	40
7.	Collar is broad,	35	7.	**Collar is broad or divided,**	35
8.	Florets is 1,	35	8.	**Florets is 1,**	35
9.	Vernation is rolled,	35	9.	**Vernation is rolled,**	**75**
10.	Glumes are shorter.	20	10.	**Glumes are shorter.**	20

† Rules were modified after initial frequency analyses of chosen values

‡ Confidence level assigned by system developers.

§ Portions of the rules which were modified appear in **bold.**

participant group was found for correctly identifying a grass species after the rules were modified.

An analysis of the frequency of the selected values for each variable by either group of participants using either identification tool showed no significant dependency on individual values for any variable. In several cases, such as the toothed value of the ligule variable for bentgrass (selected for 53% of the bentgrass specimens) and the fine value of the blade width variable for zoysiagrass (selected 94%), the identified variable value was quite different than the one provided in the original rule. In addition, many of the variables which had low CL's in the original rules were most readily identified by the participants. For example, the original CL for the vernation-rolled zoysiagrass were 35 but was selected 82% of the identifications.

One of the most prominant findings of this investigation was the relatively poor performance in the identification of unknown grasses by individuals regardless of their training. Because the mean correct identification of any species by any participant group using WEEDER was less than 60%, it was not known if an expert level performance was achieved. In a subsequent study (Fermanian et al. 1989), considered over all characters, trained participants selected the correct value 59% of the time, whereas the untrained participants did so 53% of the time. No significant association was observed between participant groups and their selection ability for ligule size, sheath, blade width, collar, and pubescence when all species were considered jointly. Various programs have been developed for the identification of plant species by matching user selected values with similarity coefficients (Pankhurst 1975; Ross 1975). While these systems have generally reported similarity values of 60 to 90% ($\pm$5%) the success rate of identifying unknown species with the systems was not reported.

When using the identification key, performance was generally better from the group with plant science training, however, the frequency analysis did not indicate a significant dependance on either participant group. This difference in performance, however, was not found when the same group used WEEDER, which generally benefited either group equally. It is important to note that a significant gain in the ability of all participants to correctly identify a grass specimen was found with WEEDER over the diagnostic key, after rules were modified to maximize the support of constently chosen correct values of variables to identify the specimens examined. While the modification of a rule generally provided for the identification of specimens which were previously not identified, it also removed some specimen identifications from the group initially considered correct. Further testing of WEEDER is required to determine if the modified rules are consistent with additional grass sample and user populations.

This study brings out one important aspect to the use of expert or advisory sytems. While the use of knowledge is central to all advisory systems, the skills associated with recognizing the value of prompted variables is paramount in plant species identification. These recognition skills were probably lacking in the test population. It is necessary, therefore, to develop techniques to enhance recognition skills to further increase the effectiveness of WEEDER (Michalski 1986).

While WEEDER provided an initial test of AGASSISTANT's inferencing capabilities, other portions of the program remain untested (learning module, rule optimization, etc.). Additional efforts currently are being developed to test these functions.

Conclusions

An expert system builder that is capable of learning and improving its knowledge has been presented. Thus it has been demonstrated that sophisticated knowledge acquisition facilities are suitable for creating expert systems in the microcomputer environment. This should be of great use in disseminating this technology to the typical agricultural user who does not have access to large computers. However, A number of improvements and extensions to AGASSISTANT are possible.

- The system could incorporate new and more powerful learning subsystems. One such method is learning by analogy, in which the program acquires knowledge by comparing to similar cases it has seen in the past.
- The ideal system should be able to adjust its knowledge during the advisory session. That is, if it is told that it made an incorrect decision, it should be able to update its knowledge in light of this information.
- The current system uses a very simplistic method for combining evidence. The creator of the system should be able to state the importance of groups of conditions in addition to weighting individual conditions.
- The system could benefit from automated methods for generated explanation and other text during the advisory session.
- The learning module should be able to incorporate background knowledge. In addition, it should be able to suggest a hiearchical structure whereby input events are connected to decision classes through intermediate nodes.
- The system could be expanded to include the programs CLUSTER (Stepp, 1983), for clustering examples into categories, and ATEST (Michalski 1985), for testing the consistency and completeness of rules.

- Finally, the system could be integrated with a video system. This would enable the system to display plants and other items during the question answering phase of the advisory session

This list can help to guide further research to make AGASSISTANT a still more powerful and useful tool for agricultural decision making.

References

Dietterich, T., and R.S. Michalski. 1983. A Comparative Review of Selected Methods for Learning from Examples. In *Machine Learning: An Artificial Intelligence Approach*, ed. R.S. Michalski, J. Carbonell and T. Mitchell. Palo Alto: TIOGA Publishing Company: 41-81. .

Fermanian, T.W., M. Barkworth, and H. Liu. 1989. Ability of Trained and Untrained Individuals to Identify Morphological Characters of Immature Grasses. *Agron. J.* 81:918-222.

Gower, J.C., and J.A. Barrett. 1971. Selecting tests in diagnostic keys with unknown responses. *Nature.* 232:491-93.

Michalski, R.S. 1973. AQVAL/1 Computer implementation of a variable valued logic system VL1 and examples of its application to pattern recognition. *Proceedings of the First International Joint Conference on Pattern Recognition,* Washington, D.C.

_____. 1975. Variable valued logic and its application to pattern recognition and machine learning. In *Computer Science and Multiple Valued Logic: Theory and Applications*, ed. D.C. Rine. North Holland.

_____. 1983. A theory and methodology of inductive learning. In *Machine Learning: An Artificial Intelligence Approach*, ed. R.S. Michalski, J. Carbonell and T. Mitchell. Palo Alto: TIOGA Publishing Company.

_____. July, 1985. Knowledge repair mechanisms: Evolution vs revolution, ISG 8514, UIUCDCSF85946, Urbana, IL: Department of Computer Science, University of Illinois (and) June 1985. *Proceedings of the Third International Machine Learning Workshop.* Skytop: Rutgers University.

_____. 1986. Understanding the nature of learning. In *Machine Learning: An Artificial Intelligence Approach,* Vol. 2, ed. R.S. Michalski, J.G. Carbonell, and T.M. Mitchell. Los Altos, CA: Morgan-Kaufmann.

_____ and A.B. Baskin. August 8-12, 1983. Integrating multiple knowledge representations and learning capabilities in an expert system: The ADVISE system. *Proceedings of the 8th IJCAI.* Karlsruhe, West Germany.

_____ and J.B. Larson. 1983. *Incremental Generation of VL1 Hypothesis: the underlying methodology and the description of program AQ11.* ISG 835, UIUCDCSF83905, Urbana IL: Department of Computer Science, University of Illinois.

_____ and R.L. Chilausky. 1980a. Knowledge acquisition by encoding expert rules versus computer induction from examples: A case study

involving soybean pathology. *International Journal for Man-Machine Studies*. 12:63–87.

——— and R.L. Chilausky. 1980b. 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*. 4(2):125–160.

———, J.H. Davis, V.S. Bisht, and J.B. Sinclair. 12–14 July 1982. PLANT/ds: An expert consulting system for the diagnosis of soybean diseases. *Proceedings of the 1982 European Conference on Artificial Intelligence*. Orsay, France. 133–138.

Morse, L.E. 1971. Specimen identification and key construction with time-sharing computers. *Taxon*. 20(1):269–82.

Pankhurst, R.J. 1975. Identification by matching. In Biological Identificaton with Computers. ed. R.J. Pankhurst. London & New York: Academic Press.

———, 1978. *Biological identification*. Baltimore, MD: University Park Press.

Reinke, R.E. July, 1984. Knowledge acquisition and refinement tools for the ADVISE MetaExpert System. M.S. Thesis ISG 844, UIUCDCSF84921, Urbana, IL: Department of Computer Science, University of Illinois.

Ross, G.J.S. 1975. Rapid techniques for automatic identification. In *Biological identification with computers*. ed. R.J. Pankhurst. London & New York: Academic Press.

Shurtleff, M.C., T.W. Fermanian, and R. Randell. *Controlling Turfgrass Pests*. Englewood Cliff, NJ: Prentice Hall.

Stepp, R. November, 1983. A Description and User's Guide for CLUSTER/2, A Program for Conjunctive Conceptual Clustering, Report No. UIUCDCSR831084, Urbana, IL: Department of Computer Science, University of Illinois.