ADAPTATION OF A SIMPLE FRAME-BASED SYSTEM TO ACCOMMODATE COMPLEX INTERACTION AMONG FRAMES

BY

V. R. REDDY SABBELLA

Bachelor of Technology

Nagarjuna University

Guntur, India

1983

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 1988

ADAPTATION OF A SIMPLE FRAME-BASED SYSTEM TO ACCOMMODATE COMPLEX INTERACTION AMONG FRAMES

Thesis Approved:

M J Folk

Thesis Adviser

[illegible]

Gerrit W. Cuperus

Norman N. Durham

Dean of the Graduate College

ACKNOWLEDGEMENTS

I wish to express sincere appreciation to Dr. Michael Folk for his encouragement and excellent guidance throughout my thesis. Profound thanks to Dr. Gerrit Cuperus for his suggestions and invaluable help in many ways throughout my graduate program. Many thanks to Dr. George for his advice and serving on my graduate committee.

Thanks go to Mr. Alan Stark for his cooperation and suggestions. Also thanks to Dr. Berberet, Dr. Stritzke, Dr. Young, Dr. Criswell, and Dr. Pinkston for their help in bringing this thesis.

Finally, I would like to thank the staff and the secretaries of Entomology department for their cooperation and help.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

Expert systems

Artificial intelligence is a growing branch of computer science that studies ways of enabling computers to do tasks that seem to require human intelligence. These tasks include expert problem solving, theorem proving, game playing, natural language understanding, speech recognition, and image processing. Out of these areas of AI the expert problem solving is well understood and has a wide range of practical applications. The programs used for solving the problems which require human expertise are known as expert systems. Expert systems are also sometimes called knowledge-based systems or rule-based systems.

Expert systems differ from conventional programs in the way they:

. separate the search mechanism from data and knowledge representation

. easily deal with symbolic data

. allow incorporation of uncertain or incomplete information

. explain the reasoning of a conclusion to the user

. allow easy addition of knowledge without requiring changes in the logic of the program

Ramamoorthy et al [13] give an interesting comparison of what makes AI programs different. Below is presented the table of comparison:

Feature	Expert systems	Conventional Programming
. Processing type	Symbolic	Numeric
. Technique	Heuristic search	Algorithmic
. Definition of solution steps	Not explicit	Precise
. Answers sought	Satisfactory	Optimal
. Control/Data separation	Separate	Intermingled
. Knowledge	Imprecise	Precise
. Modification	Frequent	Rare

Every expert system consists of two parts: an inference engine and a knowledge base. The knowledge base is unique to a particular domain, but the inference engine may be common to a number of domains that have similar characteristics.

The knowledge base contains the facts and the rules or knowledge relationships that embody an expert's knowledge. The facts, sometimes known as working memory, contain declarative knowledge about the particular problem being solved and the current state of affairs in the attempt to solve the problem. There are several different ways to represent these facts: first order predicate logic, frames, and semantic networks [13]. In predicate logic the declaration "Richard gave Jean a rose" might be represented in the form "Give(Richard, Jean, rose)". A frame for the same information might be

name of the frame: F1

type of frame: transfer of possession

source: Richard

destination: Jean

agent: Richard

object: rose

In the semantic network, each node contains an object and the lines between the nodes represent the relationships. The semantic network representation for the same example is shown in Figure 1.

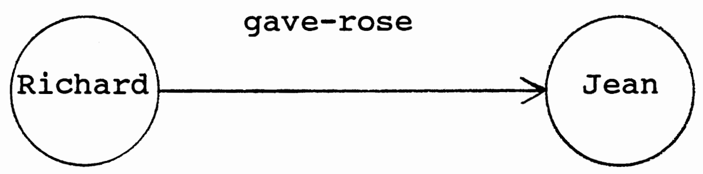

Figure 1.

The rules in a knowledge-base contain the formulas showing the relationship among several pieces of information. A typical rule looks like this:

"If it is clear and hot and muggy, then it is summer."

Here we have the three antecedents "clear", "hot", and "muggy" connected by logical AND, which, when satisfied, lead to the consequence that it is "summer".

The inference engine may use forward chaining or backward chaining to infer the rules and the facts in the knowledge base and give conclusions to the user. The inference engine should also be able to explain the reasoning of its conclusions.

In forward chaining the antecedents of each rule are evaluated and if they succeed the consequents of that rule are fired or evaluated. Reasoning in a forward chaining system is described as a "recognize-act" cycle [10]. First, the rules that can succeed, given the contents of working memory, are recognized. One rule is selected and then the action or conclusion is asserted into working memory. Then the system proceeds to the next cycle and checks again to see what rules succeed. It terminates when a desired result or conclusion is reached or when all the relevant rules are exhausted.

In backward chaining the system looks for the rule containing the required goal parameter and then sees if the antecedents of that rule succeed. If one or more of the parameters in the antecedent part are unknown it tries to find the values of those parameters in the same way as that of the goal parameter. Once the rule succeeds the goal parameter is assigned a value according to the consequent part of the rule. If this rule does not succeed the system looks for another rule which contains the goal parameter in its consequent part and repeats the above procedure with that rule. This process continues until the goal parameter is assigned some value or there are no more rules to try. This is the most common form of reasoning used in many expert systems[6].

Combinations of forward chaining and backward chaining are also used in some situations.

In the following sections we consider various approaches available for developing expert systems and the criteria to be used for selecting an expert system development tool. An overview of the problem being considered is also presented.

Tools for Building Expert Systems

There are primarily two different approaches for building an expert system. A programming language such as Lisp, Prolog, C etc. or some kind of a shell can be used. Below, we discuss the merits of using these languages and various shells.

Lisp: Traditionally, Lisp has been the language of expert systems. Most of the early expert systems were written using Lisp. Its principal data structure is the list, which is very useful in representing much of the knowledge used in expert systems. Storage space is allocated dynamically, enabling programs to be larger than they would otherwise be. The fact that both data and procedures are represented as lists makes it possible to integrate declarative and procedural knowledge into a single structure such as property list. It also makes it possible for a program to construct a procedure and then execute it. Most Lisp systems run interactively. This facilitates the development of all kinds of programs. It also makes it possible to write truly interactive programs. There exist many dialects of Lisp, varying on everything from the names of standard functions and the order of their arguments to the kinds of features provided. Common Lisp and Franz Lisp are popular versions of Lisp. MacLisp, developed at MIT, is noted for its efficiency. InterLisp of Xerox Corporation has a very sophisticated program development environment.

We can build large and highly complex knowledge-based systems using Lisp. The major disadvantage is that not many people can write programs well in Lisp. It requires drastically different way of thinking from conventional programs.

Prolog: Prolog is a recent competitor for Lisp. This is a logic programming language. This was originally developed for theorem proving. An appealing characteristic of Prolog, which adds to its increasing popularity, is that the emphasis in writing a Prolog program is on specifying or describing the nature of the problem, rather than on spelling out the steps the computer should take to solve the problem, as one is required to do in conventional languages like Pascal, Fortran, C. For this reason Prolog is considered to be a declarative or assertional language, as contrasted with conventional languages, which are usually called imperative.

A Prolog program may be viewed as a collection of logical formulas in propositional logic with a theorem (query) to be proved. Prolog provides a database facility to store facts and rules for efficient manipulation. A Prolog interpreter uses a pattern matching technique called unification to select relevant rules from the database to answer a particular query. Unification refers to substitutions of variables performed in such a way as to make two items match identically.

C: Though most expert systems have been written using either Lisp or Prolog, in recent years C is becoming more and more popular. The primary advantages of using C are its speed, low memory requirements and portability. The present trend is to use expert systems on personal computers. Since all the advantages of C mentioned above are very important for knowledge-based systems running on PCs, the usage of C is further enhanced.

Others: There is no restriction on the language to be used for developing an expert system. There are expert systems written in Fortran, Pascal, Assembler, Smalltalk, Forth etc

[10].

Expert system shells: A shell is a common program which can be adapted to any expert system of a certain type by adapting the general structure to the specifics of that system[10]. There are different varieties of commercial shells available. The capabilities and the prices of these shells vary considerably. Simple knowledge-based systems can be developed by a non-programmer using these shells. Below, a brief review of some commercially available shells follows. In the next section, the criteria to be used in selecting an expert system shell are given.

ART: ART, Automated Reasoning Tool, is a versatile tool that incorporates a sophisticated programming workbench[6]. It runs on advanced computers and workstations such as those produced by Symbolics, LMI, TI, Apollo, and VAX. ART's strong point is viewpoints, a technique that allows hypothetical nonmonotonic reasoning; in non-monotonic reasoning, multiple solutions are carried along in parallel until constraints are violated or better solutions are found. At such points, inappropriate solutions are discarded. ART is primarily a forward-chaining system with sophisticated user-defined pattern matching. ART has a flexible graphics workbench with which to create graphical interfaces and graphical simulations. ART was designed for a real-time performance. To achieve this performance, ART compiles its frame-base as well as its relational knowledge into logic-like assertions. Applications particularly suited for ART are planning/scheduling, simulation, configuration generation, and design. It is written in Lisp as well as in C. It is a product of the Los Angeles based Inference Corporation.

KEE: KEE runs on advanced AI computers. It is the most widely used programming environment for building sophisticated expert systems[6]. KEE supports windows, menus, and graphics. It contains a sophisticated frame system that allows the hierachical modeling of objects and permits multiple forms of inheritance. It supports object-oriented programming, forward-chaining, backward-chaining and hypothetical reasoning. KEE has been used for applications in diagnosis, monitoring, real-time process control, planning, design, and simulation. This is developed by California-based IntelliCorp.

Knowledge Craft: Knowledge Craft is an integration of the Carnegie Mellon version of OPS5 and of Prolog and the SRL frame-representation language. It is meant for experienced knowledge engineers and AI system builders. Knowledge Craft is capable of hypothetical reasoning. It can also support graphics-based simulation. It is used for planning, scheduling and process control. It is a product of Carnegie Group of Pittsburgh.

PC+: PC+ is an attempt to provide on a personal computer many of the advanced features found in more sophisticated tools such as KEE. Thus PC+ utilizes frames with attribute inheritance, and rules. PC+ supports both backward-chaining and forward-chaining. It is written in Lisp and incorporates user-friendly interfaces. It supports graphics and access to the popular dbase II and III database packages. It is a product of Texas Instruments, Inc..

VP Expert: VP expert is designed for personal computers. It is very simple, low cost(less than $150) and easy to use. It is written in C and can interface with the database package dbase III+ and supports windows and a limited amount of graphics. It can also access spread sheet files in the form of Lotus 123 and any executable file. VP Expert uses frames which contain rules and goals. It does not directly support attribute inheritance from frame to frame. It provides a mechanism for storing the facts to a data file and retrieving the facts from a data file. This facility can be intelligently used to inherit attributes from one frame to another frame. It uses backward-chaining as its reasoning process. It is a product of Paperback Software International.

Criteria for Selecting an Expert System Shell

Gevarter[6] identifies the following attributes to consider for selecting an expert system shell: cost, rule or size limit, function capabilities, speed, ease of learning, interfaces to other software, portability, documentation, training, company support and user satisfaction. Below, we present a review of the major function capabilities. Not all the tools are good for all kinds of functions. Each one is best suited for a particular kind of application.

Classification: This is the function most commonly addressed by expert systems. Classification refers to selecting an answer from a fixed set of alternatives on the basis of information that has been input. Below are some subcategories of classification.

- Interpretation of measurements.
- Diagnosis.
- Debugging, treatment, or repair. These functions refer to taking actions or recommending measures to correct an adverse situation that has been diagnosed.
- Use adviser. These systems depend both on the goals of the user and the current situation in suggesting what to do next. Use advisors are helpful in guiding users through procedures in other domains such as auto repair and piloting aircraft.

Design and synthesis: Design and synthesis refers to configuring a system on the basis of a set of alternative possibilities. The expert system incorporates constraints that the system must meet as well as guidance for steps the system must take to meet the user's objectives.

Intelligent assistant: Here the emphasis is on having a system that, depending on user needs, can give advice, furnish information, or perform various subtasks.

Prediction: Prediction refers to forecasting what will happen in the future on the basis of current information. This forecasting may depend upon experience alone, or it may involve the use of models and formulas.

Scheduling: Scheduling refers to time ordering a given set of tasks so that they can be done with the resources available and without interfering with each other.

Planning: Planning is the selection of a series of actions from a complex set of alternatives to meet a user's goals. It is more complex than scheduling in that tasks are chosen, not given. In many cases, time and resource constraints do not permit all goals to be met. In these cases, the most desirable outcome is sought.

Monitoring: Monitoring refers to observing an ongoing situation for its predicted or intended progress and alerting the user or system if there is a departure from the expected or usual.

Control: Control is a combination of monitoring a system and taking appropriate actions in response to the monitoring to achieve goals.

Digest of information: A system performing this function may take in information and return a new organization or synthesis. One application is the assessment of military or stock market situations on the basis of input data and corollary information.

Discovery: Discovery is similar to digest of information except that the emphasis is on finding new relations, order, or concepts. This is still a research area.

Function applications can be considered to be of two types: "surface reasoning" and "deep reasoning". In surface reasoning, no model of the system is employed; the approach taken is to write a collection of rules, each rule asserting that a certain situation warrants a certain response or conclusion. In deep reasoning, the system draws upon causal or structural models of the domain of interest to help arrive at the conclusion.

Overview of the Problem

In this thesis the focus is on simple frame-based systems having the following capabilities:

. A frame is the basic data structure for knowledge representation

. Each frame has certain goal-parameters and some if-then rules for inferencing data

. Only one frame can be active in memory at any time

. Information can be inherited to a frame is through a data file

. No predefined hierachical structure is associated with the frames

. Backward-chaining is the reasoning process used

. It can interface with a database software such as dbase

. A frame can call any executable file from anywhere within the frame

. A frame can be loaded into memory by another frame which is currently in memory or directly by the user. The new frame replaces the old one in memory.

In this thesis we take a low cost simple frame-based system such as VP Expert and analyze it thoroughly about the function capabilities that can be obtained using such system. The necessary tools to overcome some of the deficiencies of such a system will be developed.

The objective is to use an inexpensive simple frame-based system to build a reasonably complex system with several frames and complex interactions among them. A reasonably complex practical problem related to agricultural planning, more specifically alfalfa management, is taken to demonstrate how the tools being developed can be used with a simple frame-based system, VP Expert.

The crop alfalfa is a perennial crop with several cuttings each year. Its life span, yield, quality of yield and profit depend on several factors and the decisions made by the farmer [17]. Important factors are site selection, fertilization, seed selection, insect control and weed control decisions for short term and long term. The management problems are compounded by the fact that the effects of these factors interact with one another. Treatment of weeds may effect insect levels and vice versa

[1]. Also, most of the decisions will have multi-year impact on yield and profit. It requires expert knowledge to make cost effective decisions. As there are a number of factors involved multiple frames are required with strong interaction among them when an expert system for such a problem is developed.

In chapter II, we present the advantages and limitations of a simple frame-based system and look at solutions to overcome some of the limitations. In chapter III, we give some tools to overcome some of the deficiencies. Finally, in chapter IV, we summarize what we have done and look at further scope of work possible.

CHAPTER II

SIMPLE FRAME-BASED SYSTEMS

Introduction

In the last chapter, a simple frame-based system was defined as having the following capabilities:

- A frame is the basic data structure for knowledge representation
- Each frame has certain goal-parameters and some if-then rules for inferencing data
- Only one frame can be active in memory at any time
- Information can be inherited to a frame through a data file
- No predefined hierachical structure is associated with the frames
- Backward-chaining is the reasoning process used
- It can interface with a database software such as dbase
- A frame can call any executable file from anywhere within the frame
- A frame can be loaded into memory by another frame which is currently in memory or directly by the user. The new frame replaces the old one in memory.

A simple frame-based system differs from a frame-based shell in the sense that it does not support any hierachical structuring of the knowledge base and alternative search strategies.

In this chapter a subjective assessment of the function capabilities that can be obtained using such a simple frame-based system is given. The commercial shell VP Expert fits very well into this definition of a simple frame-based system. VP Expert also has been selected as the development tool for the agricultural planning problem being considered. Therefore, a brief review of how VP Expert operates is given in the next section. The advantages and limitations of such a simple shell and some possible solutions to overcome some of its deficiencies are discussed in the subsequent sections.

VP Expert

The way VP Expert operates is as follows. The basic data structure for knowledge representation is a frame. A typical frame in VP Expert looks as shown in Figure 2. The words written using capital letters are VP Expert reserved words. This frame consists of an ACTIONS block, some RULEs with an IF part and a THEN part, the "statements" ASK and CHOICES and the "clauses" MENU, FIND, WHILEKNOWN, GET, RESET and DISPLAY. The keyword UNKNOWN is used to indicate whether a given variable is assigned some value. VP Expert statements generally contain information pertinent to the

```
ACTIONS
      MENU the_wine, ALL, wines, wine
      FIND the_wine
      MENU the_price, the_wine = wine, wines, price
      FIND the_price
      WHILEKNOWN wine
      GET the_wine = wine AND the_price = price, wines,ALL
      RESET message
      FIND message
      END
      FIND option;
 RULE 1
 IF   wine <> UNKNOWN
 THEN
      message = displayed
      DISPLAY "
 Your choice, a {wine}, is priced at ${price}.
 It has a rating of {rating}.~"
 ELSE
      message = none;
 ASK the_wine: "For which wine do you want information?";
 ASK the_price: "For the selected wine, which price do you
 choose?";
 ASK option: "Do you want anything else ?"
 CHOICES option: Yes, No
```

Figure 2.

consultation. They are independent of the knowledge-base. Clauses, on the other hand, are not independent of the knowledge-base. They always occur as a part of the ACTIONS block or the conclusion of a rule.

The ACTIONS block consists of the keyword ACTIONS followed by one or more clauses and ends with a semicolon. This is a required element of a knowledge-base. It defines the "goals" of the consultation and the sequence of their solution. In other words, the actions block tells the inference engine what it needs to find out, and in what order. This is accomplished with the FIND clauses that instruct VP-Expert to find the value or values of one or more "goal variables". Whenever the VP Expert encounters a FIND clause it tries to find the value of the goal variable from the rules in the same frame by using backward-chaining. If none of the rules can assign a value to the goal variable, then it sees if there is any ASK statement associated with that variable. If there is an ASK statement for that variable, it then prompts the user for the value of the variable with the prompt given in the ASK statement. If the variable has a CHOICES statement, the choices in the CHOICES statement are displayed and the user can select from only those choices in that case. It also supports assigning confidence factors to the variable's values.

In addition to FIND clauses, the ACTIONS block may also contain other clauses specifying database operations, spreadsheet operations, arithmetic calculations and a variety of other tasks.

The MENU clause is used to give a menu of choices from a database file to the user. The first MENU clause in Figure 2 displays all the different names in "wine" field in "all" the records of the dbase III+ file "wines.dbf" as a menu to the user. The wine that the user selects is made the value of the variable "the_wine". Every thing inside the WHILEKNOWN..END loop is executed as long as the value of the variable following WHILEKNOWN is known. The RESET clause makes the value of the variable following it UNKNOWN. The DISPLAY clause displays the text inside the double quotes following it. The curly parenthesis are used to display the value of a variable.

Each RULE must have a unique label following the keyword RULE. The IF part of a RULE contains the antecedent rules using the logical operators AND and OR and the relational operators =, >, >=, <, <=, and <>. The THEN part of a RULE contains the consequents which may include any statement or clause.

VP Expert supports several other statements and clauses not shown in Figure 2. It has the clauses SAVEFACTS, and LOADFACTS, respectively for saving all the known facts during a consultation into a data file and for loading all the facts from a data file. It also provides a clause, CHAIN, to call any other frame by the frame currently in memory. Only one frame can be in memory at any given time. When frame i calls frame j, frame j is loaded into memory and frame i will be no longer in memory.

Advantages and Limitations

There are several advantages of using such a simple frame-based system. (1) It is inexpensive (less than $150). (2) It is very easy to learn and use for non-programmers as it does not have any complicated data-structures and multiple paradigms. With the system that supports several inferencing paradigms, it is difficult to present a user interface that is as easy to understand and use as a system whose design is based upon a single approach [11].(3) A simple frame-based system does not require any programming in a complicated language such as Lisp that is difficult for novices to learn. (4) Any computationally oriented modules can be separately written using a language such as C to enhance the speed of consultation time of the expert system. (5) Interfacing with the popular software packages such as dbase and Lotus is extremely simple. Therefore, it can be used as a front end of a database to intelligently update and search the database as needed.

Another major advantage is that a frame can be used by several frames as there is no hierachical structure imposed on the frames. Therefore, it offers more flexibility in terms of the interaction that may be made possible among different frames when compared to systems that use hierachical structuring of frames. Associated with this freedom and flexibility is some price. The developer of the expert system must explicitly take care of calling a frame when required and returning from the called frame to the proper place in the calling frame. Since a frame may be used as a subframe by more than one frame, it avoids any necessity for creating duplicate frames. This results in greater consistency of the knowledge-base and less storage requirements. Thus, this kind of a simple frame-based system is very desirable when some of the frames need to be shared by several frames. This is very much the case in the crop alfalfa management problem.

Shown in Figure 3 are some of the different frames needed for the alfalfa management expert system. The names with no extension are knowledge-bases. The names with a .dbf and .exe extension are the dbase and executable files respectively. An arrow from frame i to frame j indicates that frame i may need to call frame j zero or more times when frame i is consulted. Some of the frames may be combined into a single frame at the cost of understandability and efficiency. Dividing a large problem into several small frames has advantages. Notable among them are speedy development as more than one person can start working on different frames, less memory requirements, and easy maintainability. It can be observed from Figure 3 that

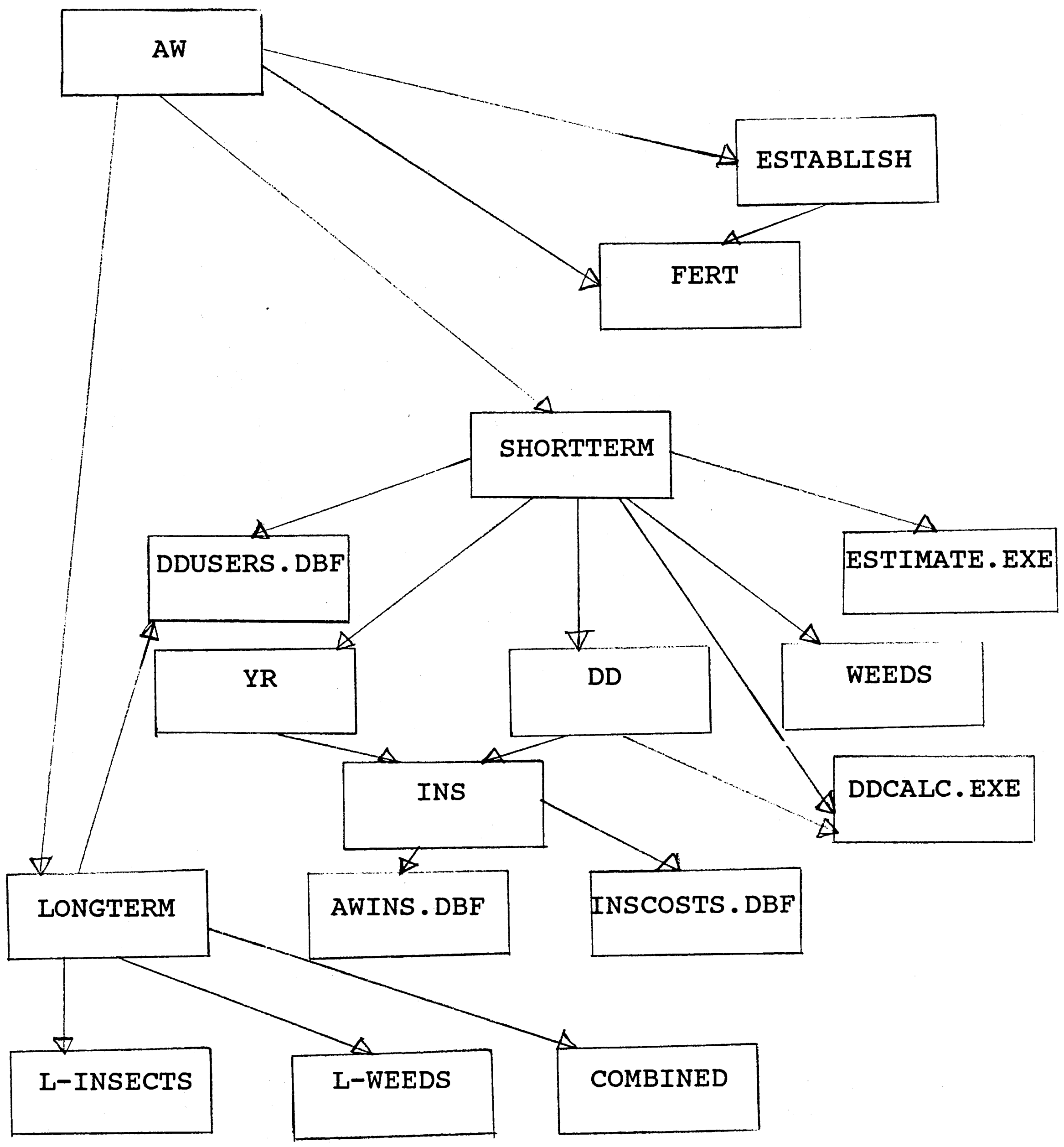
AW
ESTABLISH
FERT
SHORTTERM
DDUSERS.DBF
ESTIMATE.EXE
YR
DD
WEEDS
INS
DDCALC.EXE
LONGTERM
AWINS.DBF
INSCOSTS.DBF
L-INSECTS
L-WEEDS
COMBINED

Figure 3.

the frame INS may be called from three different sources SHORTERM, YR, and DD. Similarly, the frame FERT needs to be shared by the frames ESTABLISH and AW. In order to make the interaction among different frames easier and error-free some solutions are suggested in the next section.

There are some major limitations of this kind of a simple frame-based system. There are two levels of knowledge: meta-level and object-level. Meta-level knowledge consists of meta-rules. This knowledge permits a higher degree of intelligence by allowing the system to make deductions about itself. Meta-level knowledge also determines the most efficient strategy of operation the object-level can take. Object-level knowledge is the expertise that a knowledge base contains in the form of rules, frames, and variables. Without meta-level knowledge, the object-level knowledge has a fixed strategy of operation. When knowledge bases are small, a fixed strategy of operation poses no efficiency or performance problems. VP Expert does not provide any meta-level control as to which rules have to be tried before others. Therefore, when knowledge bases are large and several rules qualify for the given facts VP Expert can not work efficiently.

VP Expert does not support any clauses that can impose a desired range restriction on a numeric variable. Another limitation is that it is not easy to solve even a simple recursive problem such as finding whether person X is an ancestor of person Y using VP Expert.

It is also extremely difficult to support hypothetical reasoning. Hypothetical reasoning, in effect, requires creating and maintaining multiple knowledge bases during a consultation. When a situation suggests several interpretations or outcomes, virtual copies of the knowledge base are to be generated to pursue subsequent reasoning about the possibilities.

The alfalfa management problem does not require any recursive or hypothetical reasoning. It needs several frames of moderate size, usually less than 100 rules per frame. Hence meta-level rules are not of great value in this type of problem. Some of the frames need to be shared by more than one frame. Thus this kind of agricultural planning problems can exploit the simplicity and cost effectiveness of the simple frame-based systems.

Some Solutions

The afore-mentioned problem of a frame having to call several other frames can be solved by using two flags with each frame and one global data file. Solution to this problem is described briefly here and covered in detail in chapter III.

The basic idea being used is very similar to what a compiler does when a program makes a subroutine call. The called frame is analogous to a subroutine. The calling frame is analogous to the main program or calling routine.

The two flags being used are called the parent flag and the returns flag. Frame X is called the parent of frame Y iff frame X calls frame Y. The parent relationship between frames is defined only during the consultation time. There is no static relationship among the frames. Let us assume that the frame currently under consideration is X and Y is the parent frame of X. The flag parent-X is assigned the frame name "Y". This flag has to be set by the calling frame and should be saved in a data file just before it calls the frame X. Once the goal parameters in the frame X are assigned some values, frame X utilizes the parent-X flag to determine its parent.

The second flag returns-X, contains the number of times frame X called other frames after it was called by frame Y. It is equal to zero or UNKNOWN when frame X is called and is incremented by 1 every time frame X calls another frame. It becomes UNKNOWN when all the goals of the frame X are achieved. This flag is required to be able to return the control back to the point of calling. This is very much similar to saving the instruction pointer and other information before calling a subroutine and restoring the instruction pointer and rest of the information before returning to proper place in a conventional program.

The problem of restricting the values a numeric variable can take to a desired range can be solved by using two additional rules and one variable for each numeric variable. This is illustrated with the help of an example in Chapter III.

CHAPTER III

Tools For Extending a Simple Frame-based System

Introduction

In this chapter, we provide some tools to enhance the capabilities of a simple frame-based system, specifically the shell VP Expert. In the next section, we discuss some potential problems when a frame calls several other frames and how they should be tackled.

Calling Frames

VP Expert does not automatically call the required frames and return to the proper place in the calling frame. The expert system developer needs to explicitly take care of these functions. A possible solution, which uses two flags, the parent flag and the returns flag with each frame, was suggested in chapter II. Now we illustrate how this method can be successfully used with the help of an example.

Consider a hypothetical situation consisting of 6 frames A, B, C, D, X, and Y as shown in Figure 4. Here frame A has two potential parents X and Y, and it needs to call three other frames B, C, and D. Let us assume that frame A has a goal parameter goal_A that should be inferred as shown in Figure 5. The goal parameter goal_A depends on the parameters x1, x2, and x3. The parameter x1 depends on the values of the parameters in frame A before calling the frames B, C, or D. The parameters x2 and x3 have to be inferred after calling the frame B, and the frames C and D respectively. One possible solution to perform the function in Figure 5 using VP Expert is shown in Figure 6.

The following changes and additions can be observed in Figure 6. The additions are eight rules, one data file "file_x", and two flags for each of the six frames X, Y, A, B, C, and D. The returns flags of the frames X, Y, B, C, and D do not appear in Figure 6. The equations for the parameters x1, x2, and x3 in RULE 1 of Figure 6 are replaced by the three FIND clauses in RULE 1 of Figure 6. It works in the following way. The first time that frame A is consulted or called by an other frame it loads all the facts in the data file "file_x" (which may be null) and then looks for the consequent parts of the rules that assign a value to the goal parameter goal_A. It finds goal_A in rule 1. Now it determines whether all the conditions of rule 1 are satisfied. Assuming that they are satisfied, it proceeds further as described below.

Now the subgoal of frame A becomes x1. The system finds x1 in rule 2 and the antecedent Returns_A = UNKNOWN is true and therefore the value of x1 is inferred according to rule 2. The equation for x1 in rule 2 should be the same as that in rule 1 of Figure 6. Once x1 is inferred, it tries to find x2 and fires rule 3. Since frame B needs to be called to find out x2, the parent flag of the frame B, parent_B, is set to A. Because it is the first time frame A is calling any other frame the returns flag of frame A, Returns_A, is set to 1. Now, after setting the flags, the facts must be saved in the data file "file_x" and then frame B has to be called. Observe that the parameter x2 has not been assigned any value at this stage. Frame B infers all its goals, resets its parent to UNKNOWN and saves the facts in "file_x" and transfers control back to frame A. Frame A starts execution all over again beginning at the ACTIONS block. It loads the facts from the file "file_x" and tries to find its goal parameter goal_A. Rule 1 is fired again. Since the value of x1 is already known, the clause FIND x1 is ineffective. Now, it infers the value of x2 by firing rule 4 and comes back to rule 1 and tries to infer the value of x3. At this stage, the first rule whose antecedents are satisfied and can assign some value to the parameter x3 is rule 5. Therefore rule 5 is fired. The frame C is called in the same way that frame B was called after setting the flags parent_C to A and Returns_A to 2. When control comes back to frame A, it starts execution all over again beginning from the ACTIONS block and fires rule 1. Since x1 and x2 are known, the clauses FIND x1 and FIND x2 are not effective at this stage. Now rule 6 is fired and frame D is called after setting the appropriate flags and saving the facts. When control returns to frame A, it first infers the value of the parameter x3 and then the value of the goal parameter goal_A using the rules 7 and 1.

Once all the goals of the frame A are inferred, it has to pass the control back to its parent. Passing the control back to the parent is achieved through the clause FIND parent in ACTIONS block and the rules 100 and 101.

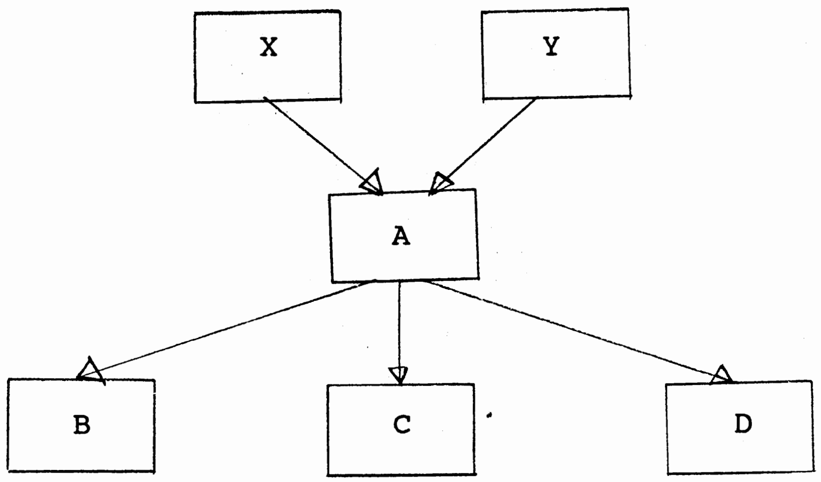

Figure 4.

```
ACTIONS

        .

        .

        FIND goal_A;
 RULE 1
 IF  <conditions>
 THEN
     x1 = f(parameters before invoking frames B, C and D)
     x2 = f(parameters after invoking frame B)
     x3 = f(parameters after invoking frames C and D)
     goal_A = f(x1,x2,x3);
```

Figure 5.

```
ACTIONS
     LOADFACTS file_x

     .

     .

     FIND goal_A
     FIND parent;
RULE 1
IF
     <conditions>
THEN
     FIND x1
     FIND x2
     FIND x3
     goal_A = f(x1,x2,x3);


RULE 2
IF
     Returns_A = UNKNOWN
THEN
     x1 = f(current parameters);


RULE 3
IF
     Returns_A = UNKNOWN
THEN
     parent_B = A
     Returns_A = 1
     SAVEFACTS file_x
     CHAIN B
     x2 = something;


RULE 4
IF
     Returns_A = 1
THEN
     x2 = f(current parameters);


RULE 5
```

```
IF

     Returns_A = 1

THEN

     parent_C = A

     Returns_A = 2

     SAVEFACTS file_x

     CHAIN C

     x3 = something;


RULE 6

IF

     Returns_A = 2

THEN

     parent_D = A

     Returns_A = 3

     SAVEFACTS file_x

     CHAIN D

     x3 = something;


RULE 7

IF

     Returns_A = 3

THEN

     x3 = f(current parameters);


RULE 100
```

```
IF   parent_A = X
THEN RESET parent_A
     SAVEFACTS file_x
     CHAIN X
     parent = null;

RULE 101
IF   parent_A = Y
THEN RESET parent_A
     SAVEFACTS file_x
     CHAIN Y
     parent = null;
```

Figure 6.

Range Restriction For Numeric Variables

VP Expert currently does not support any statement for restricting the range of values a numeric variable may be allowed to take. This can be achieved using one addtional variable and two additional rules. Shown in Figure 7 is an example to restrict the range of values the variable sl can take to 0 to 30.

Observe that we need to create an extra variable sl1 and two addtional rules to impose range restriction on sl.

```
ACTIONS
     FIND info;
RULE 1
IF   sl <= 10 AND sd < 5
THEN .
     .
     info = found;
RULE 2
IF   sl = UNKNOWN
THEN
     WHILEKNOWN range_sl
          RESET range_sl
          RESET sl1
          FIND sl1
          FIND range_sl
     END
     sl = (sl1)
     RESET sl1;
RULE 3
IF   sl1 < 0 OR sl1 > 30
THEN range_sl = not_okay
     DISPLAY "Error in stem length value. Please enter
again.";
ASK sl1: "What is the stem length in inches ?";
ASK sd: "What is the stem density ?";
```

Figure 7.

CHAPTER IV

SUMMARY AND CONCLUSIONS

Summary

In chapter I, a brief overview of expert systems was presented. The key differences between an expert system and a conventional program were outlined. Then, various tools that can be used for the development of expert systems and the criteria to be used in selecting an expert system development tool were discussed. It was observed that cost, rule or size limit, function capabilities, speed, ease of learning, interfaces to other software, portability, documentation, training, company support, and user satisfaction are important criteria to be used.

A simple frame-based system was defined and the commercial shell VP Expert was found to fall into this definition of a simple frame-based system. It was observed that simple frame-based systems are inexpensive and easy to use for non-programmers. The objective was to develop some tools to help novices to build reasonably complex systems with a simple frame-based system. A reasonably complex practical problem related to agricultural planning, more specifically alfalfa management, was taken and implemented using VP Expert.

In chapter II, a brief review of how VP Expert operates was presented. The advantages and limitations of such a simple frame-based system were discussed in detail. The primary advantages are that it has low cost, is easy to use, does not require programming in a difficult to learn language, provides flexibility for sharing frames, and is easy to interface with the database packages DBASE III+ and LOTUS. One major disadvantage is that it does not support any hierachical structuring of knowledge bases and therefore the user must explicitly take care of calling a required frame and returning to the calling frame to the proper place. It also does not support imposing range restriction for numerical variables. Another limitation is that it does not support meta-level control and therefore when knowledge bases are large it can not work efficiently. It was also observed that it can not easily support problems requiring recursive reasoning. It was further observed that the alfalfa crop management problem being considered in this thesis does not require any meta-level control or recursive reasoning. Some solutions for calling frames and imposing range restrictions on numeric variables were presented.

In chapter III, the implementation details of the proposed solutions were explained with the help of examples. The alfalfa management problem is successfully implemented using the tools that are developed. Three frames shortterm, dd, and ins are enclosed in the appendix.

Conclusions

The capabilities of the shell VP Expert in combination with the tools that were developed were found adequate for the alfalfa management problem considered in this thesis. When knowledge bases are large some kind of meta-level control is needed to make the functioning of the system efficient. VP Expert does not support any meta-level control. Providing meta-level control to VP Expert enhances its usability. Though some tools are developed to make complex interaction among frames possible, it would be a lot easier if the shell itself can take care of returning from the called frame to the calling frame to the point of calling.

BIBLIOGRAPHY

1. Berberet, R., OSU Alfalfa Teleconference Biology And Control Of The Alfalfa Weevil, Proceedings of Alfalfa Satellite Teleconference II and III, January 20 and 27, 1987

2. Bobrow, D.G., Mittal, S., Stefik, M.J., Expert Systems: Perils and Promise, CACM 29, 9(1986), 880-894.

3. Charniak, E., McDermott, D., Introduction to Artificial Intelligence, Addison-Wesley Publishing Company, 1985.

4. Denning, P.J., Towards a Science of Expert Systems, IEEE Expert 1, 2(1986), 80-83.

5. Genesereth, M.R., Ginsberg, M.L., Logic Programming, CACM 28, 9(1985), 933-941.

6. Gevarter, W.B., The Nature and Evaluation of Commercial Expert System Building Tools, Computer 20, 5(1987), 24-41.

7. Fikes, R., Kehler, T., The Role of Frame-Based Representation in Reasoning, CACM 28, 9(1985), 904-920.

8. Hartley, R.T., CRIB: Computer Fault-finding Through Knowledge Engineering, Computer 17, 3(1984), 76-83.

9. Hayes-Roth, F., Rule-Based Systems, CACM 28, 9(1985), 921-932.

10. Harmon, P., King, D., Expert Systems, John Wiley & Sons,

1985.

11. Mettrey, W., An assessment of tools for building KB systems, AI Magazine 8, 4(1987), 81-88.

12. O'keefe, R.M., Balci, O., Smith, E.P., Validating expert System Performance, IEEE Expert 2, 4(1987), 81-89.

13. Ramamoorthy, C. V., Shekhar, S., Garg, V., Software Development Support for AI Programs, Computer 20, 1(1987), 30-40.

14. Rich, E., The Gradual Expansion of Artificial Intelligence, Computer 17, 5(1984), 4-12.

15. Rich, E., Artificial Intelligence, Mcgraw-Hill Book Company, 1983.

16. Williams, C., Expert Systems, Knowledge Engineering, and AI Tools - An overview, IEEE Expert 1, 4(1986), 66-70.

17. Ward, C. E., Economics Of Alfalfa Production, Proceedings Alfalfa Management Satellite Teleconference II and III, January 20 and 27, 1987.

18. _____, VP-Expert Rule-Based Expert System Development Tool, Paperback Software International, 1987.

19. _____, Personal Consultant Plus, Texas Instruments Incorporated, 1986.

20. _____, Peter Hart talks about Expert Systems, IEEE Expert 1, 1(1986), 96-99.

APPENDIX

SOFTWARE IMPLEMENTATION OF ALFALFA MANAGEMENT PROBLEM

APPENDIX

FRAME: SHORTTERM

```
EXECUTE;
RUNTIME;
ENDOFF;
ACTIONS
     LOADFACTS awdata1
     FIND the_name
     CLOSE userdd
     GET the_name = lname,userdd,ALL
     old_date = ((lyear)*10000 + (lmonth)*100 + (lday))
     FIND new_date
     SAVEFACTS awdata1
     FIND the_choice
     FIND frame
     CHAIN shortterm;

RULE 0
IF new_date = UNKNOWN
THEN WHILEKNOWN range_date
          RESET range_date
          RESET year
          RESET month
          RESET day
          FIND year
          FIND month
          FIND day
          new_date1 = ((year)*10000 + (month)*100 + (day))
          FIND range_date
     END
     new_date = (new_date1);


RULE 1
IF the_choice = dd_based_recommend
THEN choice = dd
     parent_dd = shortterm
     CLOSE userdd
     WHILEKNOWN dd_flag
          RESET dd_flag
          RESET dd_option
          FIND dd_option
          FIND dd_flag
     END
     FIND cnl
     FIND sl
     FIND nl
     SAVEFACTS awdata
     CHAIN dd
     frame = dd;

RULE 2
IF   the_choice = yld_based_recommend
THEN choice = yr
     parent_yr = shortterm
     FIND sl
     FIND nl
     SAVEFACTS awdata
     CHAIN yr
     frame = yr;

RULE 3
IF   the_choice = insecticide_select
THEN choice = ins
     parent_ins = shortterm
     SAVEFACTS awdata
     CHAIN ins
     frame = ins;

RULE 4
IF   the_choice = Weed_recommend
THEN frame = weeds
     choice = weeds
     parent_weeds = shortterm
     SAVEFACTS awdata
     CHAIN weeds;

RULE 5
IF   the_choice = calc_day_degrees
THEN call ddcalc
     frame = none;

RULE 6
IF   the_choice = yield_estimate
THEN call estimate
     frame = none;

RULE 7
IF   the_choice = Exit
THEN RESET the_choice
     CHAIN aw
     frame = aw
     parent = aw;
```

```
RULE 8
IF   dd < 0 OR dd > 1200 OR dd < (old_dd)
THEN DISPLAY "Error in day degree information. Please enter
again. "
     range_dd = not_okay;

RULE 9
IF   old_date > ((new_date1))
THEN range_date = not_okay
     DISPLAY "Invalid date. Please enter again.";

RULE 10
IF   month < 0 OR month > 5 OR day < 0 OR day > 31
     OR year < 88 OR year > 99
THEN range_date = not_okay
     DISPLAY "Error in date. Please enter again.";

RULE 11
IF   dd_option = calc_individual_dd
THEN call ddcalc
     dd_flag = known;

RULE 12
IF   dd_option = get_dd_from_file
THEN CLOSE userdd
     GET the_name = lname, userdd,ALL
     dd = (old_dd)
     D I S P L A Y   "   Y o u r   d a y   d e g r e e s   a s   o f
{lmonth}/{lday}/{lyear} are : {old_dd} "
     dd_flag = known;

RULE 13
IF   dd_option = update_dd_in_file
THEN WHILEKNOWN range_dd
          RESET range_dd
          RESET dd
          FIND dd
          FIND range_dd
     END
     GET the_name = lname, userdd, ALL
     FIND update
     dd_flag = known;

RULE 14
IF   lname = UNKNOWN
THEN lname = (the_name)
     lyear = (year)
     lmonth = (month)
     lday = (day)
     old_dd = (dd)
     APPEND userdd
     update = done
```

```
ELSE
     lyear = (year)
     lmonth = (month)
     lday = (day)
     old_dd = (dd)
     PUT userdd
     update = done;

RULE 16
IF   dd > 540 AND cnl_text = decreased_10_or_more
THEN cnl = -10;

RULE 17
IF   dd > 540 AND cnl_text = within_10
THEN cnl = 0;

RULE 18
IF   dd > 540 AND cnl_text = increased_10_or_more
THEN cnl = 10;

RULE 19
IF   sl1 < 0 OR sl1 > 30
THEN range_sl = not_okay
     DISPLAY "Error in stem length. Please enter again.";

RULE 20
IF   nl1 < 0 OR nl1 >75
THEN range_nl = not_okay
     DISPLAY  "Error  in  number  of  larvae.  Please  enter
again.";

RULE 21
IF   sl = UNKNOWN
THEN WHILEKNOWN range_sl
          RESET range_sl
          RESET sl1
          FIND sl1
          FIND range_sl
     END
     sl = (sl1)
     RESET sl1;

RULE 22
IF   nl = UNKNOWN
THEN WHILEKNOWN range_nl
          RESET range_nl
          RESET nl1
          FIND nl1
          FIND range_nl
     END
     nl = (nl1)
     RESET nl1;
```

```
ASK the_name:"Enter your last name  please :";
ASK year :"Enter the year (Eg:88):";
ASK month:"Enter the month (Eg:4):";
ASK day:"Enter the day (Eg:15):";

ASK the_choice: "What would you like ?";
CHOICES  the_choice:Calc_Day_degrees,  DD_based_recommend,
Yld_based_recommend,
Yield_estimate, Insecticide_select, Weed_recommend, Exit;

ASK dd_option: "Enter your option:";
CHOICES  dd_option:Calc_individual_DD,  Get_DD_from_file,
Update_DD_in_file, Enter_DD[not_saved], Continue;

ASK cnl_text: "What is the change in number of larvae since
last sample ?";
CHOICES  cnl_text:  Decreased_10_or_more,  Within_10,
Increased_10_or_more;

ASK dd:"Enter the day degrees :";
ASK sl1: "Enter the stem length in inches:";
ASK nl1: "Enter the number of larvae per 30 stems:";
```

FRAME: DD

```
ENDOFF;
EXECUTE;
RUNTIME;
ACTIONS
     LOADFACTS awdata
     FIND goal_dd
     RESET Returns_dd
     SAVEFACTS awdata
     FIND parent;

RULE 0.0
IF   Returns_dd = UNKNOWN
THEN FIND Recommendation
     Returns_dd = 1
     FIND dd_flag1
     FIND dd_flag2
     FIND Insecticide
     goal_dd = achieved;

RULE 0.1
IF   Returns_dd = 1
THEN goal_dd = achieved
     RESET goal_dd;

RULE 0.2
IF   Recommendation = Spraying
THEN Insecticide = needed
     parent_ins = dd
     SAVEFACTS awdata
     CHAIN ins;

RULE 0.3
IF   Recommendation = Spray_or_harvest and
     dd >= 540
THEN Insecticide = Needed
     parent_ins = dd
     SAVEFACTS awdata
     CHAIN ins;

RULE 0.4
IF   parent_dd = shortterm
THEN SAVEFACTS awdata
     CHAIN shortterm
     parent = shortterm;

RULE 0.5
IF   dd < 150
THEN Recommendation = Resampling_in_7_days;

RULE 1
IF   dd >= 150 AND dd < 240 AND
     sl <= 2 AND nl >= 13
THEN
     Recommendation = Spraying;

RULE 2
IF   dd >=150 AND dd < 240 AND
     sl =3 AND nl >=20
THEN
     Recommendation = Spraying;

RULE 3
IF   dd >=150 AND dd < 240 AND
     sl = 4 AND nl >=30
THEN
     Recommendation=Spraying;

RULE 4
IF   dd >=150 AND dd < 240 AND
     sl = 5 AND nl >=35
THEN
     Recommendation=Spraying;

RULE 5
IF   dd >=150 AND dd < 240 AND
     sl >= 6 AND sl <= 8 AND nl >=40
THEN
     Recommendation=Spraying;

RULE 6
IF   dd >=150 AND
     dd < 240 AND
     sl =3 AND nl >=20
THEN
     Recommendation=Spraying;

RULE 7
IF   dd >=150 AND dd < 240 and
     sl <= 2 and nl < 13
THEN Recommendation = Resampling_in_5_to_7_days;

RULE 8
IF   dd >=150 AND dd < 240 and
     sl = 3 and nl < 20
THEN Recommendation = Resampling_in_5_to_7_days ;
```

```
RULE 9
IF   dd >=150 AND dd < 240 and
     sl = 4 and nl < 30
THEN Recommendation = Resampling_in_5_to_7_days ;

RULE 10
IF   dd >=150 AND dd < 240 and
     sl = 5 and nl < 35
THEN Recommendation = Resampling_in_5_to_7_days ;

RULE 11
IF   dd >=150 AND dd < 240 and
     sl <= 8 and nl < 40
THEN Recommendation = Resampling_in_5_to_7_days;

RULE 12
IF   dd >= 240 AND dd < 290 AND
     sl <= 3 AND nl >= 10
THEN Recommendation = Spraying;

RULE 13
IF   dd >= 240 AND dd < 290 AND
     sl >= 4 AND SL<=6 AND nl >= 15
THEN Recommendation = Spraying;

RULE 14
IF   dd >= 240 AND dd < 290 AND
     sl >= 7 AND SL<=8 AND nl >= 20
THEN Recommendation = Spraying;

RULE 15
IF   dd >= 240 AND dd < 290 AND
     sl >= 9 AND SL<= 11 AND nl >= 25
THEN Recommendation = Spraying;

RULE 16
IF   dd >= 240 AND dd < 290 AND
     sl <= 3 AND nl < 10
THEN Recommendation = Resampling_in_5_to_7_days;

RULE 17
IF   dd >= 240 AND dd < 290 AND
     sl >= 4 AND sl <=6  AND nl < 15
THEN Recommendation = Resampling_in_5_to_7_days;

RULE 18
IF   dd >= 240 AND dd < 290 AND
     sl >= 7 AND sl <= 8  AND nl < 20
THEN Recommendation = Resampling_in_5_to_7_days;
```

```
RULE 19
IF   dd >= 240 AND dd < 290 AND
     sl >= 9 AND sl <= 11  AND nl < 25
THEN Recommendation = Resampling_in_5_to_7_days;

RULE 20
IF   dd >= 290 AND dd < 340 AND
     sl <= 3 AND nl >= 10
THEN Recommendation = Spraying;

RULE 21
IF   dd >= 290 AND dd < 340 AND
     sl = 4  AND nl >= 18
THEN Recommendation = Spraying;

RULE 22
IF   dd >= 290 AND dd < 340 AND
     sl >= 5 AND sl <= 7  AND nl >= 25
THEN Recommendation = Spraying;

RULE 23
IF   dd >= 290 AND dd < 340 AND
     sl >= 8 AND sl <= 14  AND nl >= 30
THEN Recommendation = Spraying;

RULE 24
IF   dd >= 290 AND dd < 340 AND
     sl <= 3 AND nl < 10
THEN Recommendation = Resampling_in_5_to_7_days;

RULE 25
IF   dd >= 290 AND dd < 340 AND
     sl = 4  AND nl < 18
THEN Recommendation = Resampling_in_5_to_7_days;

RULE 26
IF   dd >= 290 AND dd < 340 AND
     sl >= 5 AND sl <= 7  AND nl < 25
THEN Recommendation = Resampling_in_5_to_7_days;


RULE 27
IF   dd >= 290 AND dd < 340 AND
     sl >= 8 AND sl <= 14  AND nl < 30
THEN Recommendation = Resampling_in_5_to_7_days;

RULE 28
IF   dd >=340 AND dd < 389 AND
     SL >=6 and sl <=7 and nl >=25
THEN Recommendation = Spraying;
```

```
RULE 29
IF   dd >=340 AND dd < 389 AND
     SL >=8 and sl <=10 and nl >=30
THEN Recommendation = Spraying;

RULE 30
IF   dd >=340 AND dd < 389 AND
     SL >=11 and sl <=16 and nl >=35
THEN Recommendation = Spraying;

RULE 31
IF   dd >=340 AND dd < 389 AND
     SL >=17 and sl <=17 and nl >=25
THEN Recommendation = Spraying;

RULE 32
IF   dd >=340 AND dd < 389 AND
     SL >=6 and sl <=7 and nl >=14 and nl <=24
THEN Recommendation = Resampling_in_3_to_5_days;

RULE 33
IF   dd >=340 AND dd < 389 AND
     SL >=8 and sl <=10 and nl >=14 and nl <=29
THEN Recommendation = Resampling_in_3_to_5_days;

RULE 34
IF   dd >=340 AND dd < 389 AND
     SL >=11 and sl <=11 and nl >=14 and nl <=34
THEN Recommendation = Resampling_in_3_to_5_days;

RULE 35
IF   dd >=340 AND dd < 389 AND
     SL >=12 and sl <=16 and nl >=17 and nl <= 34
THEN Recommendation = Resampling_in_3_to_5_days;

RULE 36
IF   dd >=340 AND dd < 389 AND
     SL >=17 and sl <=17 and nl >=18 and nl <= 39
THEN Recommendation = Resampling_in_3_to_5_days;

RULE 37
IF   dd >= 340 and dd <389 and
     sl >= 6 and sl <= 11 and nl <= 13
THEN Recommendation = Resampling_in_5_to_7_days;

RULE 38
IF   dd >= 340 and dd <389 and
     sl >= 12 and sl <= 16 and nl <= 16
THEN Recommendation = Resampling_in_5_to_7_days;
```

```
RULE 39
IF   dd >= 340 and dd <389 and
     sl >= 17 and sl <= 17 and nl <= 17
THEN Recommendation = Resampling_in_5_to_7_days;

RULE 40
IF   DD >= 390 AND DD <= 539 AND
     SL >= 11 AND SL <= 12 AND NL >= 20
THEN RECOMMENDATION = Spraying;

RULE 41
IF   DD >= 390 AND DD <= 539 AND
     SL >= 12 AND NL >= 25
THEN RECOMMENDATION = Spraying;

RULE 42
IF   DD >= 390 AND DD <= 539 AND
     SL >= 14 AND SL <= 15 AND NL >= 30
THEN RECOMMENDATION = Spraying;

RULE 43
IF   DD >= 390 AND DD <= 539  AND
     SL >= 16 AND SL <= 17 AND NL >= 35
THEN RECOMMENDATION = Spraying;

RULE 44
IF   DD >= 390 AND DD <= 539 AND
     SL >= 18 AND NL >= 40
THEN RECOMMENDATION = Spraying;

RULE 45
IF   DD >= 390 AND DD <=539 AND
     SL >= 11 AND SL <=12 AND NL >= 8 AND NL <= 19
THEN RECOMMENDATION = Resampling_in_3_to_5_days;

RULE 46
IF   DD >= 390 AND DD <=539 AND
     SL = 13 AND NL >= 8 AND NL <= 24
THEN RECOMMENDATION = Resampling_in_3_to_5_days;

RULE 47
IF   DD >= 390 AND DD <=539 AND
     SL >= 14 AND SL <=15 AND NL >= 14 AND NL <= 29
THEN RECOMMENDATION = Resampling_in_3_to_5_days;

RULE 48
IF   DD >= 390 AND DD <=539 AND
     SL = 16 AND NL >= 14 AND NL <= 34
THEN RECOMMENDATION = Resampling_in_3_to_5_days;
```

```
RULE 49
IF   DD >= 390 AND DD <=539 AND
     SL = 17 AND NL >=18 AND NL <= 34
THEN RECOMMENDATION = Resampling_in_3_to_5_days;

RULE 50
IF   DD >= 390 AND DD <=539 AND
     SL = 18 AND NL >= 18 AND NL <= 39
THEN RECOMMENDATION = Resampling_in_3_to_5_days;

RULE 51
IF   DD >= 390 AND DD <= 539 AND
     SL >= 11 AND SL <=13 AND NL <=7
THEN RECOMMENDATION = Resampling_in_5_to_7_days;

RULE 52
IF   DD >= 390 AND DD <= 539 AND
     SL >= 14 AND SL <=16 AND NL <=13
THEN RECOMMENDATION = Resampling_in_5_to_7_days;

RULE 53
IF   DD > 750
THEN RECOMMENDATION = Stop_sampling;

RULE 54
IF   DD >= 540 AND DD <= 750  AND
     SL > 15 AND NL > 35 AND CNL = -10
THEN RECOMMENDATION = Spray_or_harvest;
RULE 55
IF   DD >= 540 AND DD <= 750  AND
     SL > 15 AND NL > 30 AND CNL > -10  and CNL < 10
THEN RECOMMENDATION = Spray_or_harvest;

RULE 56
IF   DD >= 540 AND DD <= 750  AND
     SL > 15 AND NL > 25 AND CNL = 10
THEN RECOMMENDATION = Spray_or_harvest;

RULE 57
IF   DD >= 540 AND DD <= 750 AND
     SL > 15 AND NL >=18 AND NL <= 34 AND CNL <= -10
THEN RECOMMENDATION = Resampling_in_3_to_5_days;

RULE 58
IF   DD >= 540 AND DD <= 750 AND
     SL > 15 AND NL >=14 AND NL <= 29 AND CNL > -10 and CNL
< 10
THEN RECOMMENDATION = Resampling_in_3_to_5_days;

RULE 59
IF   DD >= 540 AND DD <= 750 AND
     SL > 15 AND NL >=8 AND NL <= 24 AND CNL >= 10
```

```
THEN RECOMMENDATION = Resampling_in_3_to_5_days;

RULE 60
IF   DD >= 540 AND DD <=750 AND
     SL > 15 AND NL >=0 AND NL <=17 AND CNL <= -10
THEN RECOMMENDATION = Resampling_in_5_to_7_days;

RULE 61
IF   DD >= 540 AND DD <=750 AND
     SL > 15 AND NL >=0 AND NL <=13 AND CNL >= -10 AND CNL
<= 10
THEN RECOMMENDATION = Resampling_in_5_to_7_days;

RULE 62
IF   DD >= 540 AND DD <=750 AND
     SL > 15 AND NL >=0 AND NL <=7 AND CNL >= 10
THEN RECOMMENDATION = Resampling_in_5_to_7_days
ELSE
     DISPLAY "  "
     DISPLAY "Your input does not make sense."
     RECOMMENDATION = NOTHING;

RULE 63
IF   DD >= 540 AND Recommendation = spray_or_harvest
THEN dd_flag1 =1
     CLS
     DISPLAY "The recommendation is to spray or harvest."
     WOPEN 1, 3,1,17,44,2
     WOPEN 2, 3,45,17,30,2
     WOPEN 3, 20,1,3,70,2
     ACTIVE 1
     DISPLAY "CUT EARLY"
     DISPLAY "Advantages: Tall alfalfa is VERY tolerant of
defoliating insects like alfalfa weevils.
If you cut early you will :
   1. Save insecticide costs
   2. Increase probability beneficial organisms such as
fungi and parasites."
     DISPLAY "Drawbacks:
   1. Your samples only indicate large larvae. There maybe
a substantial number of smaller larvae present.
   2. Substantial damage to regrowth under the windrow is
possible.
   3. There will be a yield reduction of up to 20 percent
simply due to cutting^G      early."
     ACTIVE 2
     DISPLAY "    APPLY INSECTICIDE"
     DISPLAY "Advantages:
   1. Risks due to weevils feedings under the windrow are
reduced.
   2. Cutting on time will increase yield by up to 20
percent."
```

```
     DISPLAY "   "
     DISPLAY "Disadvantages:
   1. Costs of insecticide application.
   2.  Potential  benefit  from  beneficial  insects  is
eliminated.
   3. Fungus may have taken care of weevil problems."
     ACTIVE 3
     FIND SPRAY_OPTION
     WCLOSE 1
     WCLOSE 2
     WCLOSE 3
ELSE dd_flag1 = 2
     DISPLAY "Based  on  day  degrees,  stem  height,  and  the
number of larvae present,"
     DISPLAY "We recommend {Recommendation}. "
     LOCATE 10,0
     DISPLAY "press any key to continue..~";

RULE 64
IF   dd_flag1=1 AND SPRAY_OPTION = Harvest_early
THEN  DISPLAY  "Be  careful.  Make  sure  that  you  have
understood the situation well."
     LOCATE 15,0
     DISPLAY "press any key to continue..~"
     dd_flag2 = 1;

ASK spray_option: "Do you want to spray or harvest early
?";
CHOICES spray_option: Spray, Harvest_early;
```

FRAME: INS

```
EXECUTE;
RUNTIME;
ENDOFF;
ACTIONS
     LOADFACTS  awdata
     FIND goal_ins
     RESET Returns_ins
     SAVEFACTS awdata
     FIND parent;


RULE 0.0
IF   Returns_ins = UNKNOWN
THEN WHILEKNOWN  range_days
          RESET range_days
          RESET days_to_harvest
          FIND days_to_harvest
          FIND range_days
     END
     FIND days_to_harvest
     FIND dd_days_okay
     CLS
     RESET ins_choice
     FIND ins_choice
     CLS
     RESET insecticide
     FIND insecticide
     CLS
     DISPLAY "Recommended insecticides are:"
     DISPLAY  "(R  means  restricted  use.   N  means  no
restriction on the use of insecticide"
     DISPLAY  "------------------------------------
-----------------------------------"
     DISPLAY "   Insecticide      Restr.-    quantity --
wait      ---- cost"
     DISPLAY "                    R - restr.   (AI/acre)
period          per"
     DISPLAY "                   N - none
(days)          acre ($)"

     DISPLAY  " ---------------------------------------
-----------------   "
```

```
     I = 1
     J = 0
     WHILEKNOWN insecticid
          FIND ins_flag
          RESET ins_flag
          I = (I+1)
     END
     DISPLAY " ----------------------------------------------
---------------------- "
     FIND ins_flag2
     RESET ins_flag2
     goal_ins = achieved
     RESET goal_ins;

RULE 0.01
IF   parent_ins = yr
THEN ins_flag2 = 1
     FIND ins_flag3
     RESET ins_flag3
     RESET cost_option
ELSE
     ins_flag2 = 1
     DISPLAY"press any key to continue..~";

RULE 0.02
IF   cost_option = own_cost
THEN WHILEKNOWN range_cost
          RESET range_cost
          RESET cost
          FIND cost
          FIND range_cost
     END
     ins_flag3 = something
ELSE
     ins_flag3 = something
     RESET ins_choice
     FIND ins_choice
     CLOSE awins
     I=0
     WHILEKNOWN ins_flag1
       RESET ins_flag1
       FIND ins_flag1
       I = (I+1)
      END;

RULE 0.1
IF   J < 10
THEN GET days_to_harvest >= minwait AND days_to_harvest <=
maxwait AND the_insect = insect,awins, ALL
     CLOSE INSCOSTS
     the_insecticide = (insecticid)
     GET the_insecticide = insname, inscosts, ALL
     cost = ((cost_perlb)*(quantity))
     FORMAT cost, 5.2
     DISPLAY "{I}.  {PRINTTEXT}              {cost} "
     ins_flag = 1
     J = (J+1)
ELSE DISPLAY "press any key to display more insecticides.
~"
     GET days_to_harvest >= minwait AND days_to_harvest <=
maxwait AND the_insect = insect,awins, ALL
     CLOSE INSCOSTS
     the_insecticide = (insecticid)
     GET the_insecticide = insname, inscosts, ALL
     cost = ((cost_perlb)*(quantity))
     FORMAT cost, 5.2
     DISPLAY "{i}.  {PRINTTEXT}             {cost} "
     J = 0
     ins_flag = 1;

RULE 0.2
IF   parent_ins = yr
THEN FIND i_cost
     SAVEFACTS awdata
     CHAIN yr
     parent = yr;

RULE 0.3
IF   parent_ins = shortterm
THEN SAVEFACTS awdata
     CHAIN shortterm
     parent = shortterm;

RULE 0.4
IF   parent_ins = dd
THEN SAVEFACTS awdata
     CHAIN dd
     parent = dd;

RULE 0.5
IF   i < (ins_choice)
THEN GET days_to_harvest >= minwait AND days_to_harvest <=
maxwait AND the_insect = insect,awins, ALL
     CLOSE INSCOSTS
     the_insecticide = (insecticid)
     GET the_insecticide = insname, inscosts, ALL
     cost = ((cost_perlb)*(quantity))
     FORMAT cost, 5.2
     ins_flag1 = 1;



RULE 0.6
```

```
IF   days_to_harvest < 0 OR days_to_harvest > 75
THEN range_days = not_okay
     DISPLAY  "Error  in  days  to  harvest.  Please  enter
again.";

RULE 0.7
IF   cost <=0 OR cost > 20
THEN range_cost = not_okay
     DISPLAY  "Error  in  insecticide  cost.  PLease  enter
again.";

RULE 0.8
IF   parent_ins = dd AND days_to_harvest <= 7 AND dd < 450

THEN DISPLAY "Something is wrong in your input. You can not
have your day^Gdegrees less than 450 if you are going to
harvest in less than a week time."
     LOCATE 20,0
     DISPLAY "Press any key to continue..~"
     CHAIN aw
     dd_days_okay = no;

RULE 1
IF   CHOICE = dd OR CHOICE = yr
THEN THE_INSECT = ALFALFA_WEEVIL
     ins_choice = something ;

RULE 2
IF   CHOICE = ins
THEN
     MENU the_insect,ALL,awins,insect
     FIND the_insect
     ins_choice = something;

RULE 3
IF   days_to_harvest >=15 and days_to_harvest <= 28 and
     the_insect = alfalfa_weevil
THEN insecticide = parathion
     DISPLAY "We recommend a short residual compound such
as PARATHION, or MALATHION
or a reduced rate of LORSBAN or FURADAN because:

         1. In most years, Weevil egg lay is over by this
point in the year.
         2. With good application, most small and large
larvae will be killed
            and this application should give 2 weeks of
protection."
     LOCATE 15,0
     DISPLAY  "press  any  key  to  display  possible
insecticides ...~";
RULE 4
```

```
IF   days_to_harvest >=29 and days_to_harvest <=45 and
     the_insect = alfalfa_weevil
THEN insecticide = Furadan_or_Lorsban
     DISPLAY "We recommend a long residual compound such as
FURADAN or LORSBAN."
     DISPLAY "  "
     DISPLAY "Your other option is to use a short residual
compound such as parathion.
If you choose this option, remember that this will only
give 2-3 weeks of protection.
Since there may still be egg lay you will probably need two
applications of a short residual compound."
     LOCATE 15,0
     DISPLAY "press any key to display possible recommended
insecticides ...~";

RULE 5
IF   days_to_harvest > 45 and the_insect = alfalfa_weevil
THEN insecticide = Parathion
     DISPLAY "We recommend a short residual compound such
as PARATHION because:
     1. There may still be a late freeze that could damage
alfalfa and greatly
        reduce weevil populations.
     2. There may still be substantial egg lay after the
next 2 weeks, and
     3. A long residual compound may expire before end of
egg lay.

After using a short residual pesticide, check carefully in
2-3 weeks."
     LOCATE 15,1
     DISPLAY "press any key to display possible recommended
insecticides ...~";


ASK the_insect: "Which insect are you interested in
controling? ";
ASK days_to_harvest: "Enter the days till harvest
(approximate):";
ASK ins_choice: "Enter the number of your insecticide
choice";
ASK cost_option: "Do you want to give your own insecticide
cost or you want to pick one from the table ?";
CHOICES cost_option: own_cost, pick_from_table;
ASK cost: "What is the cost of insecticide per acre in
dollars ? ";
```

VITA 2

V. R. Reddy Sabbella

Candidate for the Degree of

Master of Science

Thesis: ADAPTATION OF A SIMPLE FRAME-BASED SYSTEM TO ACCOMMODATE COMPLEX INTERACTION AMONG FRAMES

Major Field: Computing and Information Sciences

Biographical:

Personal Data: Born in Anaparty, India, March 15, 1961, the son of China Veerreddy and Bulli Ammaye Sabbella. Married to Madhavi Sabbella on May 29, 1985.

Education: Graduated from the Board of Intermediate Education, Hyderabad, India, in May 1978; received Bachelor of Technology Degree in Mechanical Engineering from Nagarjuna University, Guntur, India, in May, 1983; received Master of Technology Degree in Industrial and Management Engineering from Indian Institute of Technology, Kanpur, India, in March, 1985; completed requirements for the Master of Science degree at Oklahoma State University in July, 1988.

Professional Experience: Computer Programmer, Entomology Department, Oklahoma State University, October, 1986, to July, 1988.