A HYBRID EXPERT SYSTEM FOR CONTROL OF WEEDS ON CROPS

By

BALAJI S. HOLUR

Bachelor of Engineering
S.J.College of Engineering
Mysore, India
1983
Master of Technology
Indian Institute of Technology
Kanpur, India
1986

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
May, 1991

This work I sincerely dedicate to

My parents,

Dr.Mayfield, & Dr.Nofziger

A HYBRID EXPERT SYSTEM FOR

CONTROL OF WEEDS ON

CROPS

Thesis Approved:

Blayne E. Mayfield

J P Chandler

D W Hoyle

Norman N Durham

Dean of the Graduate College

ACKNOWLEDGEMENTS

I wish to express sincere appreciation to Dr. Mayfield for his encouragement and advice throughout my graduate program. He not only helped me in crucial hours of my graduate program, but also helped me in broadening my views in many areas. I sincerely, dedicate this thesis to my mentor, Dr. Mayfield. Many thanks also go to Dr. Nofziger, who supported me as a student professional and helped me whenever I was in need. Thanks are also due to Dr.Chandler, who encouraged me throughout my graduate program. I fail in my duty, if I don't appreciate the efforts of Dr.Samadzadeh, who always helped me to gain my morale back by constructive criticism during my graduate program.

My parents, my friends Arun, Jayanthi, Nidi, Aarathi, encouraged me and supported me throughout my graduate studies and made me reach the end goal. My friend Ravi, who gave me lot of his creative time and was so patient with me, I extend many hugs. My sisters Sujatha, Vasudha, and my friend Uma provided moral support and believed in my abilities. Last, but not the least, my wife, Rukmini who is giving me moral support in finishing my graduate studies.

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Artificial Intelligence can be defined as the search for general computational models of human intelligence [9,21,23]. The two commercially viable sectors of Artificial Intelligence are the knowledge-based systems and the subset of these systems called expert systems [1].

An Expert System attempts to solve problems in specialized domains using the same techniques as human experts in a specialized field would. To be successful, an expert system should not only attempt to solve problems in its domain of expertise, but also justify its solution, produce explanations whenever desired, and degrade gracefully in face of incomplete information.

One aim of expert systems is to emulate the human intelligence in a machine, based on anecdotal information and stored as heuristic rules. These heuristics, along with facts about the problem domain, constitute the knowledge base of the problem domain. One of the other goals of expert systems is to keep the knowledge about the problem domain (knowledge-base) separate from the inference mechanism. The inference mechanism, broadly speaking, is the way in which the rules and facts in the knowledge base are

used in a particular context to reach a solution (or goal).

1.1. Heuristics in search

Most of the problems in AI can be viewed in terms of a state space. Finding an optimal solution can be thought of as searching this space until an optimal solution state is found, if it exists. Usually, the problem domain is extremely large, and an undirected or blind search leads to combinatorial explosion [3].

On the other hand, the nature of many problems is such that there does not exist a definite procedure or algorithm that can produce a solution by traversing minimum or even small number of states. We must therefore rely on heuristics or "rules of thumb". Heuristic models provide information about the likelihood that any specific node is a better choice to try next than another [10]. In other words, heuristics are simply some rules that qualify the possibility that a search is proceeding in the right direction. Heuristics do not guarantee that an optimal solution will be found in minimal time. In fact, they do not guarantee a solution at all. But use of heuristics is a definitely better choice over a blind search, in that they reduce the search state space.

There is now a theory of heuristics [21]. Much work has been done in AI regarding the development of heuristic for several of the areas and building systems that incorporate heuristics.

Knowledge acquisition, one of the initial phases in the development of an expert system, is the process of getting these

heuristic rules from an expert (in the problem domain) and storing the information in a computer. In the development of an expert system, the knowledge acquisition activities are generally confined to the domain of expertise [21] in which the expert system has to solve problems.

1.2. Knowledge Representation

Another common underlying feature of expert systems is the separation of knowledge about the problem domain from the inference methodology. This specialized knowledge is stored in a knowledge base. A knowledge base differs from a database in that, the knowledge base contains common sense or general world knowledge, specialized or domain knowledge, and the context of interaction, in the form of rules and facts. A database, on the other hand, consists of only facts about the problem domain.

Yet another common underlying feature of expert systems is that the knowledge representation constituting the knowledge base of an expert system conventionally follows the declarative form rather than the procedural form [21]. In other words, the knowledge base of an expert system consists of a collection of rules which are not executed sequentially (as in procedural forms), but which 'fire' only as and when appropriate conditions are met. This form of declarative representation of knowledge permits easy understandability and maintenance besides being context independent.

1.3. Inference Mechanism

The inference mechanism forms the brain of an expert system. When a problem is presented to this mechanism, it tries to arrive at a solution, by using the rules and facts about the problem domain (available in the Knowledge base), as applicable in the particular context. If required, it may also instantiate queries in order to further define the context.

The inference mechanism may use either or both of two problem-solving methodologies, namely, forward chaining or backward chaining. In forward chaining, the inference mechanism works from an initial state to the goal state [6]. In backward chaining, it works from a defined goal state towards a solution state which satisfies the goal state.

The inference mechanism usually brings with it the ability to backtrack and try other alternatives. When a particular search path turns out to be unsuccessful in the search of the goal, it retraces it's path to a point where it can take a different search path, if such a path exists. Many shells are commercially available in the market today. They may use forward chaining or backward chaining or implement both to afford flexibility. Some of the commercially available shells are CxPERT, VP-EXPERT, Knowledge-Pro, etc.. Choice of a shell generally depends on the application, constraints on portability, and interface with other higher level procedural languages.

1.4. Simulation

According to Herbert Maisel et al, " Simulation is a numerical technique for conducting experiments on a digital computer, which involves certain types of mathematical and logical models that describe the behavior of a system over an extended period of time."[11].

Simulation employs various representations in order to model some aspect of an uncertain world, with the model being formed as a piece of computer software. Simulation model assumes that we can describe a system in terms acceptable to computer system. In order that a system is acceptable, a clear system state description is required. This allows a system to be characterized by a set of variables and each combination of variable values represents a unique state or condition of the system. With these representations, manipulation of variable values simulates movement of the system from state to state [19]. So some well defined operating rules are required to observe the dynamic behavior of a model, when the above said variable values are changed. In this way simulation models can be used for design, procedural analysis, and performance assessment.

Changes in the state of a system can occur continuously over time or at discrete instants of time [25,26]. Accordingly, simulation is classified into discrete simulation and continuous simulation. In discrete simulation, dependent variables change discretely at specified points in simulated time, referred to as event times [22]. In continuous simulation, the dependent variables of the model may change continuously over simulated

time.

In the present work a simulation package developed by Nofziger et.al [8], called CHEMRANK is used. CHEMRANK simulation is used to predict the environmental impact of the herbicides used to control the weds in fields.

1.5. Review of previous work

Though several researchers are working on the area of Artificial Intelligence applications in different fields, it is only recently that an effort was made to combine the functions of expert systems and simulation. One such successful attempt is an application of expert systems to simulation software [24]. In this application, a simulation program called Air Combat Evaluation Machine (ACEM) was used [24]. This program was primarily designed for analyzing avionics systems on airborne interceptors engaged in air-to-air combat. A user interface was designed and implemented for the ACEM tactics expert system.

Knowledge-based systems can be used as an intelligent front-end or back-end systems [2]. In front-end systems, knowledge base systems have been integrated into applications to remove the burden of syntax and format and [2] apply domain-specific knowledge to request user information. In some cases, these front-end systems can provide best guesses when the user is unable to provide required input. In case of backend systems, the output of a simulation model is interpreted by the system. Used in this way, knowledge-based systems can decipher the results of simulation and apply them to a user-specific

situation.

Though the work done so far in this field stresses mostly the different ways of integrating the knowledge-based systems with simulation, not much work has been done on the human factors involved in the design [10]. The evaluation of the user interface of an interactive system was first attempted by Ben Shneiderman[10]. The author presents design issues, offers experimental evidence, and makes reasonable recommendations. Accommodating human diversity is a design issue; For example, a right-handed male designer with computer training and a desire for rapid interaction using densely packed screens may have a hard time developing a successful workstation for left-handed women artists with a more leisurely and free-form work style. Understanding the physical, intellectual, and personality differences among users is vital.

Work has been carried out to create various systems with simulation models and other conventional tools as applied to agriculture [2]. This work also explains why several existing simulation models cannot be used to their full potential. Though simulation models can provide a better understanding of a real world system, it is not always possible to present parameters precisely to such models, because they cannot be precisely measured [2].

1.6. Objective and Scope of the present study

In the present work, an attempt was made to integrate a simulation model and an expert system with an application in

agriculture. The simulation package called "CHEMRANK" was used, which ranks chemicals on the basis of their groundwater polluting potential [8]. An expert system was developed to help users select herbicides for controlling weeds attacking the user's crops. These herbicides were selected based on both economical and environmental considerations. The CxPERT shell was used for developing the expert system, which also interfaced with CHEMRANK.

The key design issues considered were portability for different architecture, easy future development.

1.7. Outline of the thesis

Chapter II presents the features of the CxPERT shell used for the development of the expert system, and presents it's merits. In Chapter III, the theory and equations behind the present work are presented. The implementation details are also discussed in this chapter. In Chapter IV, the working of this expert system is presented. Finally, the conclusions, the limitations, and scope for future work are outlined.

CHAPTER II

A PERSPECTIVE OF EXPERT SYSTEMS AND SIMULATION

2.1. Introduction

In this chapter a discussion is carried out about how experts system and simulation can be used to increase the efficiency of the analysis carried out on any selected problems. The discussion is carried out in four sections.

The first part is in sections 2.2 through 2.6. In these sections, and explanation of the techniques of simulation and the similarities and differences between the expert systems and simulation is given, and an analysis of the advantages gained by combining simulation and Knowledge Based Expert System (KBES) is made. It is this concept of combination that is employed in the present research to upgrade the analysis.

The second part of the discussion begins in section 2.7. In that section advantages of using expert systems and simulation are explained. The different methods of construction and execution of the model, type of data used, analysis of the results, etc., are presented. Section 2.8 explains different methods of combining simulation and

expert systems.

Finally, in sections 2.9 and 2.10, discussions are carried out to justify the combination of the expert systems and simulation. The combination of these two is observed to produce good results [30]. The combination is basically achieved by introducing knowledge data base to simulation systems. The combination is referred to as the "Knowledge Based Simulation Systems (KBSS)". By using knowledge based simulation, the goal to a specific problem may be reached at a faster rate. This is due to the fact that in a KBES the decisive statements not only give the user optional facilities, but also they follow a defined paths based on certain heuristics to reach a goal state. On the other hand, in a simulation system, the ambiguity of whether a model behaves in a predicted manner prevails.

In order to make a comparative study between expert systems and simulation systems it is necessary to make a brief study on simulation systems and expert systems.

2.2. Simulation concepts

In Chapter I, a brief, general description of the simulation is presented. This section presents different techniques of simulation.

In its broadest sense, computer simulation is the process of designing a mathematical/logical model of a real system and experimenting with this model on a computer. If a real system can be characterized by a set of variables, with

each combination of variables representing a unique state or condition of the system, the manipulation of the variable values simulates movement of the system from state to state. A simulation experiment involves observing the dynamic behavior of a model moving from state to state in accordance with well-defined operating rules designed into the model.

Changes in the state of a system can occur continuously over time or at discrete instants in time. The discrete instants can be established deterministically or stochastically depending on the nature of model inputs. More detailed aspects of different techniques of simulation can be obtained from [19,26,30,31,32].

The nature in which the dependent variables change can be deterministic or stochastic [19]. For example, consider the simulation of the path traced by a fighter aircraft, when it sights a bomber [19]. In order to pursue the bomber, the path traced by the bomber has to be followed by the fighter aircraft. The fighter aircraft has to change its path continuously. The problem is continuous in the sense that changes in the fighter's dependent variable (i.e., its direction of flight) occurs continuously with time [19].

Similarly, consider the simulation of a bank teller [19]. Customers arrive at the bank and wait for service by a teller. If all tellers are busy, the customers wait in a queue. When a teller becomes available, a customer from the queue is serviced. After being serviced, a customer departs from the system. Customers arriving to the system when the

teller is busy, wait in a single queue in front of the teller. The total time that a customer spends in the system (which is the sum of the arrival time, time in the queue, and the service time) depends upon the number of customers in the queue and the teller status. Since the number of customers in the queue and the teller status vary discretely with respect to time, the above example is considered to be discrete in nature.

In the example of the fighter aircraft, a dependent variable is the path followed by the aircraft. In order to pursue the bomber, the aircraft has to follow a definite path, so its direction changes continuously. Since the path followed is predetermined, its direction is deterministic in nature.

However, in the example of a bank teller, the dependent variables are the number of customers in the queue at any given time and the teller status. Since the arrival rate and the service rate of customers cannot be predetermined, changes in the dependent variables (number of customers and teller status) are stochastic in nature.

In most simulations, time is the major independent variable. Other variables included in these simulations are functions of time and are dependent variables. The adjectives discrete and continuous when modifying simulation refer to the behavior of the independent variables.

Discrete simulation occurs when dependent variables change discretely only at specified points in simulated time referred to as event times. The time variable may be either

continuous or discrete in such a model, depending on whether the discrete changes in the dependent variable can occur at any point in time or only at specified points. In general the values of the dependent variables for discrete models do not change between event times. A typical response of a discrete simulation is given in figure.2.1a.

In continuous simulation the dependent variables of the model may change continuously over simulated time. A continuous model may be either continuous or discrete in time, depending on whether the values of the dependent variables are available at any point in simulated time or only at specified points in simulated time. A typical response of a continuous systems simulation is depicted in figure.2.1b.

In combined simulation, the dependent variables of a model may change discretely, continuously, or continuously with discrete jumps superimposed. The time variable may be continuous or discrete. The most important aspect of combined simulation arises from the interaction between discretely and continuously changing variables. Figure.2.1c represents a typical response of a continuous simulation with discrete time steps. This implies that it is possible to have a continuous simulation with discrete time events.

2.3. A Perspective of Expert System

As discussed in chapter I, expert systems are those systems that attempt to duplicate or imitate the results of

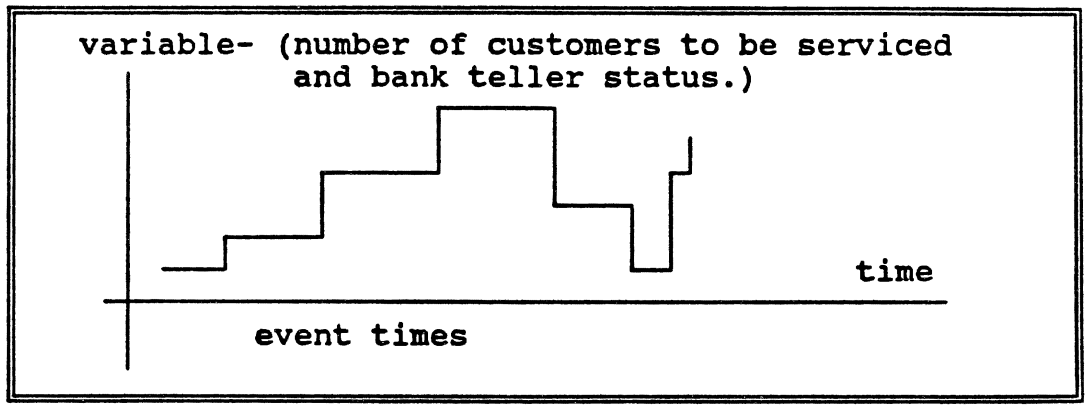

Figure.2.1a. A typical response of a discrete simulation. Event times correspond to the time at which customers arrive to the system and depart from the system in a bank teller example.

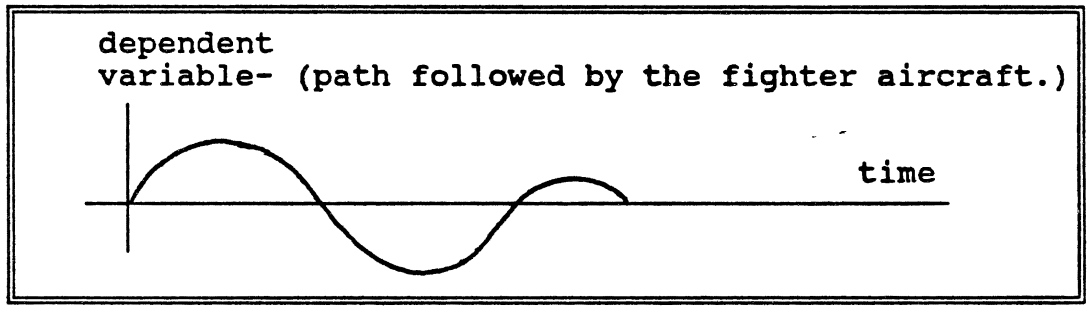

Figure.2.1b. A typical response of a continuous simulation. Value of the dependent variables may change at any point in the simulated time. The path followed by the fighter air craft is available at any instant of time.

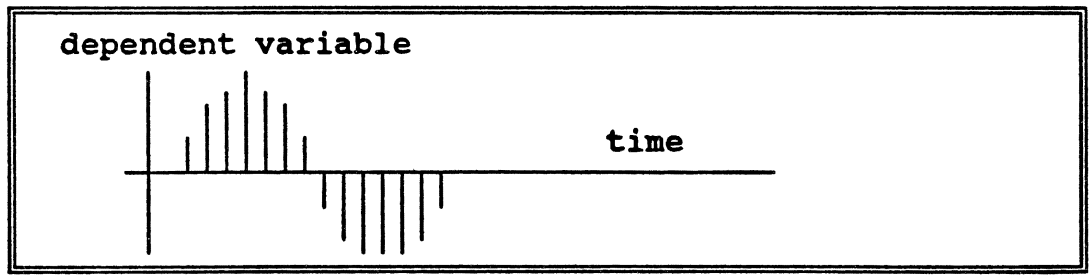

Figure.2.1c. A typical response of a continuos simulation using discrete time steps.

applying learned skills or expertise without concern for whether the processes matches those used by a human expert [1]. Expert or knowledge-based systems are designed to aggregate the experience of any number of human experts in a given field in a series of rules. These rules are then used to draw inferences and suggest to the user a course of action to deal with a given problem.

A rule is uaually a clause that contains an implication and atmost a single conclusion [1]. The literals (a literal is a predicate or inverse of a predicate) to the left of the implication sign are called the antecedents, and the literals to the right are called the conclusion. Sometimes the conclusion is an action or a suggestion. The rules are expressed as:

IF antecedent THEN conclusion.

In a typical expert system, knowledge is organized into three categories.

(1) A global data base provides the input data (that is the declarative knowledge) defining the particular problem and which keeps track of the current solution status or situation.

(2) A knowledge base which describes the expert facts and heuristics associated with the general problem domain. It is this main knowledge that takes form of rules.

(3) A control or inference structure which defines the problem solving approach or how the data and knowledge can be used to solve the problem.

At the control level the computer program makes

decisions about what question is to be answered and what control strategy to use. A control strategy is a function of the problem to be solved and several approaches can be used including:

- Forward chaining: if the solution starts from an initial set of data and conditions (the existing global data base) and moves toward some goal or conclusion, it is called model, data, event or antecedent driven. For example, consider the following two rules:

  RULE 2: IF patient P suffers from disease type A
  AND patient P exhibits symptom Z
  THEN patient P suffers from disease type B.

  RULE 1: IF patient P exhibits symptoms X AND Y
  THEN patient P suffers from disease type A

Forward chaining starts with the antecedents of a rule say, RULE 1 and attempts to establish if RULE 1 is TRUE (i.e., it moves forward through a rule.). Thus,

if patient P suffers symptoms X and Y, then it is recorded that patient P suffers from disease type A.

Rules that have disease A in their antecedent are examined next, say RULE 1, and the truths of the associated predicate, say Z, are determined.

- Backward chaining: if the desired conclusion or goal state is already known, but the path to that conclusion is not known, then working backward is called for and the model is said to be goal or expectation driven.

Considering the rules given in the example for forward chaining, backward chaining starts with a proposed

conclusion, say disease B, and finds a rule that has just this conclusion, say RULE 1. The truth of the antecedent is established by first finding rules that have elements of the antecedent as conclusions, say RULE 2, and then determining if the antecedents of those rules are true.

It is possible that an expert system control structure uses forward chaining, backward chaining, or a combination of both in reaching the goal state. However, the use of these approaches depends mainly on the type of problem under consideration.

The basic idea of this section is to highlight the advantages gained in combining expert system and simulation systems. In order to analyze the combination of the two, it is necessary to understand the similarities and the differences present in the two approaches.

The next section presents a description about the similarities between the expert system and simulation.

2.4. Similarities between Expert System and Simulation

Expert systems and simulations have many similarities. Both individually facilitate the user to achieve accuracy and expertise in reaching a particular goal.

Simulation languages and simulation models contain many of the ideas being used in AI; Some examples are: the ability of entities to store knowledge that describes their characteristics in FRAMES (FRAMES are the data structures),

to dynamically modify flow of entities through the system, to change the system based upon state variables (PATTERN INVOKED PROGRAMS), and to represent knowledge about the system in the form of a network [31].

Both, simulation and expert systems analyze expertise in a stepwise manner providing a set of rules. These rules facilitate a layman to achieve the expertise.

For example, consider a game of chess. In order to achieve expertise in the movements of pawns in this game an expert system or simulation based system can be employed. The various types of moves of pawns can be represented as rules and entered into the computer one by one. These rules facilitate a layman(who may not be aware of the moves) to achieve expertise in the movements of pawns.

Both, in simulation and in expert systems, an approximated knowledge of the consequence of an activity processed in real-world situations can be achieved. In simulation, an approximated prototype of the original is employed. This prototype is subjected to several constraints in order to obtain an idea of the behavior of the original subjected to the same constraints.

In a KBES, however, although no prototype is employed; the geometry and the behavior of the approximated prototype is fed into the computer as data. This data is subjected to specified constraints and gives the result of subjecting the process to the constraints.

2.5. Differences between Simulation Systems and Expert Systems

Even though both KBES and simulation models are concerned with the same issue - assisting humans in dealing with complex systems- they are markedly different in the way they function. This section explains the differences between an expert system and simulation systems. A KBES provides a prescription, a simulation model provides a prediction. Put another way, given a goal a KBES suggests a course of action, while a simulation model predicts the consequences of a selected course of action under some conditions. Also, a KBES may provide a rationale or an explanation for the suggested course of action.

However, construction of the model for these complex systems follow different paths. Many simulation systems follow an iterative approach in solving problems. In such an approach, a model is designed, inputs to the model are decided, and the experiment are run. Then, a second set of inputs is decided and experiments are run again, and so on.

Given a problem, many of the current simulation systems cannot decide upon an appropriate model (presuming that all classes of possible acceptable models are defined) nor do they aid in deciding how to exercise it to find an answer to our problem.

An AI-based system follows a totally different approach. Knowledge about the system (especially the description of the objects), is incorporated as data into

result to be obtained by the system) is defined in the computer program. The strategy to be used by the system to reach the goal is fed as knowledge into the computer. This knowledge is defined for a class of possible and/or acceptable models. It is the responsibility of the expert system to automatically find a model that fulfills the desired specifications and to execute an appropriate search to obtain the desired solution.

Second, in an AI-based system the data base, knowledge base and control structure should be separated so that each can be modified easily without affecting the other. Most simulation models have integrated information and control much like a conventional program. In an integrated form, the information and control are dependent upon each other.

A third difference is in the nature of the data base. Traditional data base systems require formally defined data structures and operations performed on static (at the time of data processing) data bases. This is a very rigid process that is not easily changed as requirements change.

In contrast, the structure of an AI-data base system allows an additional type of data to be collected and used. This data is symbolic, represents knowledge of facts, judgement, rules, intuition, and experience (heuristic knowledge) about a narrow problem area [26]. Heuristic data provides the rules of relationship between the elements of the traditional data in the data base and can be easily added or modified. Simulation based systems exhibit direct connectivity of data elements during the simulation process.

On the other hand, AI systems do not depend on showing the direct connectivity of data elements in the knowledge base. This allows the system to use different processes (known as the inference mechanism), to evaluate the knowledge base, interpret the data in the knowledge base, perform logical deductions and modify the knowledge base to derive information about the subject.

2.6. Characteristics of KBES and Simulation

Following are some of the characteristics of conventional simulations.

* They are numeric;
* The solution steps employed in solving a simulation based system module are highly specific and explicit. In other words they are algorithmic. The algorithm is not subjected to many decisive statements. They possess a sequential behavior.
* In simulation based system modules the data structure and the control structure are combined together, i.e., the information about an object and the strategy employed in solving the problem are both present in an integrated form.
* Models cannot do anything which is not preplanned (i.e., user must instruct the operation of the program).

An AI-based expert system on the other hand would exhibit the following characteristics.

* They can have symbolic processes.

* They can use pattern invoked search (solution steps are not explicit).
* They can have a command structure separate from knowledge domain.
* They can have all the expertise possible built into the model so that decisions by user would be minimized.
* They can have a model that would be able to learn from its own experience and modify itself as needed. In general, AI system cannot do this yet although it is theoretically possible.

2.7. Advantages of combining Expert Systems and Simulation

The following example gives some of the practical aspects achieved in combining expert systems and simulation.

Consider, a human teaching assistant who is made available to students outside of class as a resource to aid in solving assignments. The syntactic and many of the semantic errors made by students in a simulation language can be corrected by an interactive modelling facility. However, the logical errors made by students are difficult to detect and correct. The teaching assistant is employed to detect logical errors. He or she would be a person who has taken and passed that course. One of the problems associated with the use of student teaching assistants is the loss of expertise realized when they leave. A teaching assistant may spend semesters learning to provide support

assistant may spend semesters learning to provide support for a particular course. When he or she leaves the knowledge base he or she has accumulated is lost. A new teaching assistant then begins the development process again, typically rebuilding the knowledge base from the scratch and forming his own teaching philosophy toward problem solving. Consequently the quality of support provided suffers while the expertise is being acquired. In some courses an entire semester may be needed before a teaching assistant has achieved a fully acceptable level of competence.

In order to overcome this problem, a knowledge based simulation system could be used. In a KBSS, knowledge is introduced into simulation. In addition to debugging syntax errors, the KBSS could also contain knowledge to eliminate logical errors. This allows a means of capturing and retaining the expertise acquired by the teaching assistants in a permanent form. Also, it would make the accumulated knowledge readily available in useful form to the new teaching assistants and to students.

Generally, a simulation model contains different numerical models, while the knowledge base contains the semantic information necessary for the use of these models. Introducing knowledge base into a simulation model results in the following advantages:

- Accumulating knowledge into a separate component makes it easier to update the same when needed.
- The knowledge base provides logical information about the

system, whereas distinct numeric models incorporate time dependent knowledge, described in algorithmic form. The former supports the selection of problem-specific modelling methodologies.

- In a knowledge based simulation model, the user does not see the numerical aspects and the detailed conditions which guide their use. The knowledge base hides these detail and presents a compact semantic description to the user. The following section explains different ways of combining the expert systems and simulation.

2.8. Combining Simulation and Expert Systems

Combining expert system concepts with traditional simulation methodologies yields a powerful design support tool known as knowledge based simulation [31]. This approach turns a descriptive simulation tool into a perspective tool, one which recommends specific goals [31]. Ideally a KBSS system should:

* Accept a description of the problem and synthesize a simulation model by consulting an appropriate knowledge base.
* Accept a goal in the form of a set of expectations or constraints, select a model at an appropriate level of abstraction, determine the performance metrics, generate a search space of possible scenarios, execute the simulation model by controlled selection of scenarios, and finally recommend a scenario that satisfies the

stated goal [19].

* Explain the rationale behind why only certain scenarios have been explored and why it recommends a particular scenario.
* Learn from experience and disclose its behavior (perhaps by displaying significant events and what led to those events).
* Display the resultant model (i.e., one built by KBS) with a high degree of accuracy to increase user confidence.

Figure 2.2 shows a taxonomy for combining simulation and expert systems.

Perhaps the most obvious way in which the two can be combined is by embedding one within the other figures 2.2a 2.2b. It is arguable that many simulation models already use knowledge, as opposed to data. For instance, a queue priority rule is knowledge [29]. It may be pertinent to keep such rules in a knowledge base, rather than embedded in code. It may be necessary to embed a simulation within an expert system for two reasons. First, the expert system may need to run a simulation to obtain some results for the user.

Second, and more important, the expert system may use one or more time-dependent variables, and thus need a simulation.

Simulations and expert systems that are designed, developed, and implemented as separate software, in parallel, may interact [31]. A simulation model could interrogate an expert system (figure 2.2c). This may be

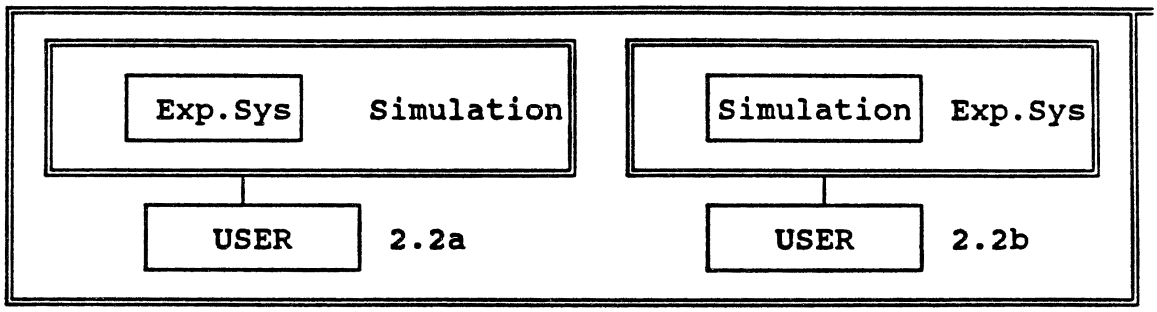

Figure.2.2a,2.2b. Embedded systems.

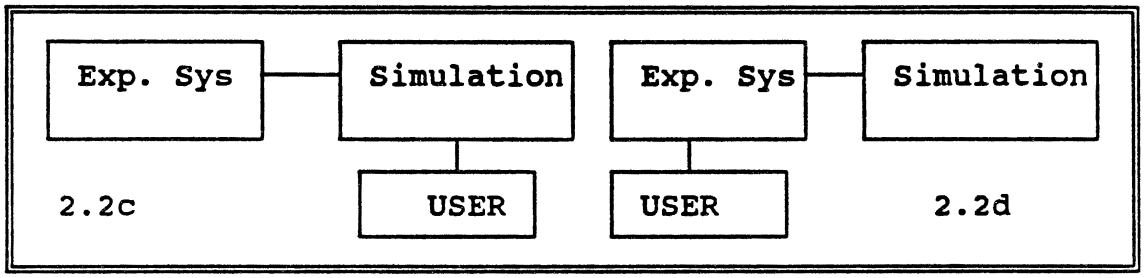

Figure.2.2c, 2.2d. Parallel Systems.

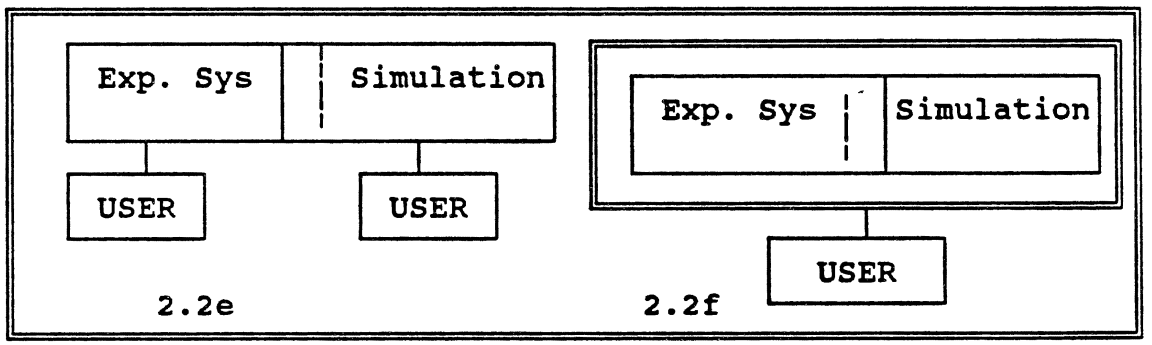

Figure.2.2e, 2.2f. Cooperative Systems.

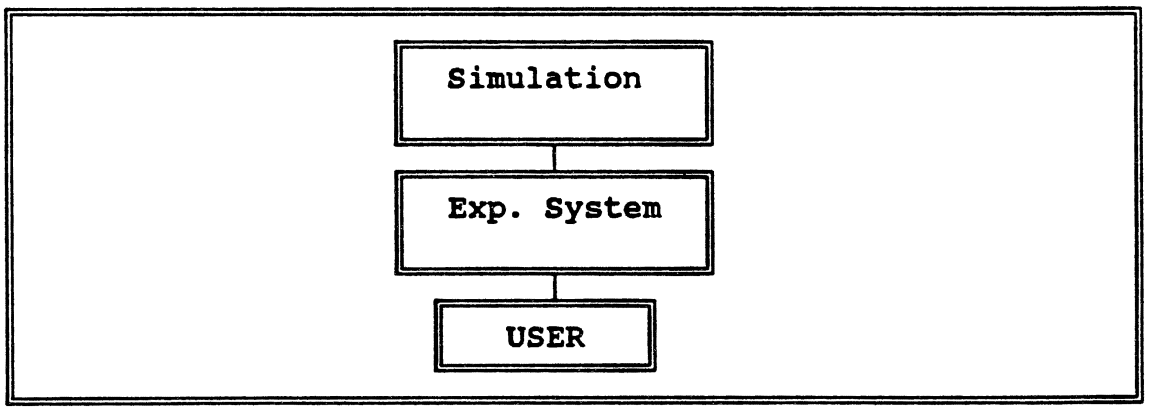

Figure.2.2g. Intelligent Front end Systems.

useful where a simulation is developed for a complex system, and an expert system already exists for part of the decision making within that system. The simulation can then access this, rather than mimic or encode the decision rules. Expert systems that execute and use the results from simulations (figure 2.2d) are of increasing interest to knowledge engineers. In addition to testing an expert system on a user or a real environment, the expert system can be tested on a simulation. Not only is development time reduced, but also testing can be more comprehensive. Further if the simulation is a valid model, then an expert system that adequately controls or responds to the simulation is, perhaps valid itself. Application domains where this approach may be useful include real-time process control, where an expert system developed to ultimately control the process can be tested on a continuous simulation of that process. In figures 2.2a through 2.2d the user of the main tool does not have direct access to the other. However, in many instances both expert system and simulation will be used together to do some task(see figure 2.2e). They may share data; in effect, the simulation and expert system will cooperate in the task.

The cooperative simulation and expert system may be surrounded by a larger piece of software as in figure 2.2f. Each may be a part of a simulation environment or part of a large decision support system used directly by decision makers. New tools based on both simulation and knowledge-based methods fall into this category.

One of the most important application areas for knowledge-based methods is Intelligent Front Ends (IFEs) (see figure.2.2g). This is an expert system that sits between a simulation package and a user, generates the necessary instructions or code to use the package following a dialogue with the user, and interprets and explains results from the package. Simulation program generators perform some of the functions of IFEs, although they do not actually execute the simulation and interpret the results [32].

Useful intelligence includes:

* Dialogue handling (a natural language interface or at least user-directed free format input).
* A model of the target package, so that some decisions can be made by the IFE rather than referred to the user.

One such attempt at combining expert systems and simulation is made present work, to develop a knowledge based simulation to control the weeds attacking the crops. The expert system here sits between the simulation package, called "CHEMRANK", and the user, generating necessary instructions, getting the necessary conditions to run the software through a dialogue with the user. Section 2.8 gives an explanation about the use of the system.

2.9. Knowledge Base Simulation System for control of weeds

In the present work, an attempt is made to explore this knowledge based simulation technique applied to find the herbicides to control weeds attacking the crops and the effect of these herbicides on ground water. The knowledge based simulation model is divided into three phases. This is depicted in figure.2.3. As explained in the figure, phase-1, consists of selection of herbicides based on certain input conditions from the user. Also, based on the herbicides selected and the soil chosen by the user, the soil properties, recharge rate, and the herbicide properties are passed to the simulation package.

Phase-2 consists of running the simulation package to rank these herbicides based on the attenuation factor method, as explained in section.4.3 of chapter IV. Phase-3, consists of the analysis of the ranks obtained by simulation and presenting the same to the end user. It also, tells the user, whether or not a second application of herbicides is required to control weeds for a given input condition. The different parameters involved in selecting the herbicides, the parameters involved in ranking of these herbicides, and the analysis of the results are explained in chapter IV.

As seen from the figure 2.3, the present system here is termed as Intelligent Front End System. Because, the expert system here sits between the simulation package, called

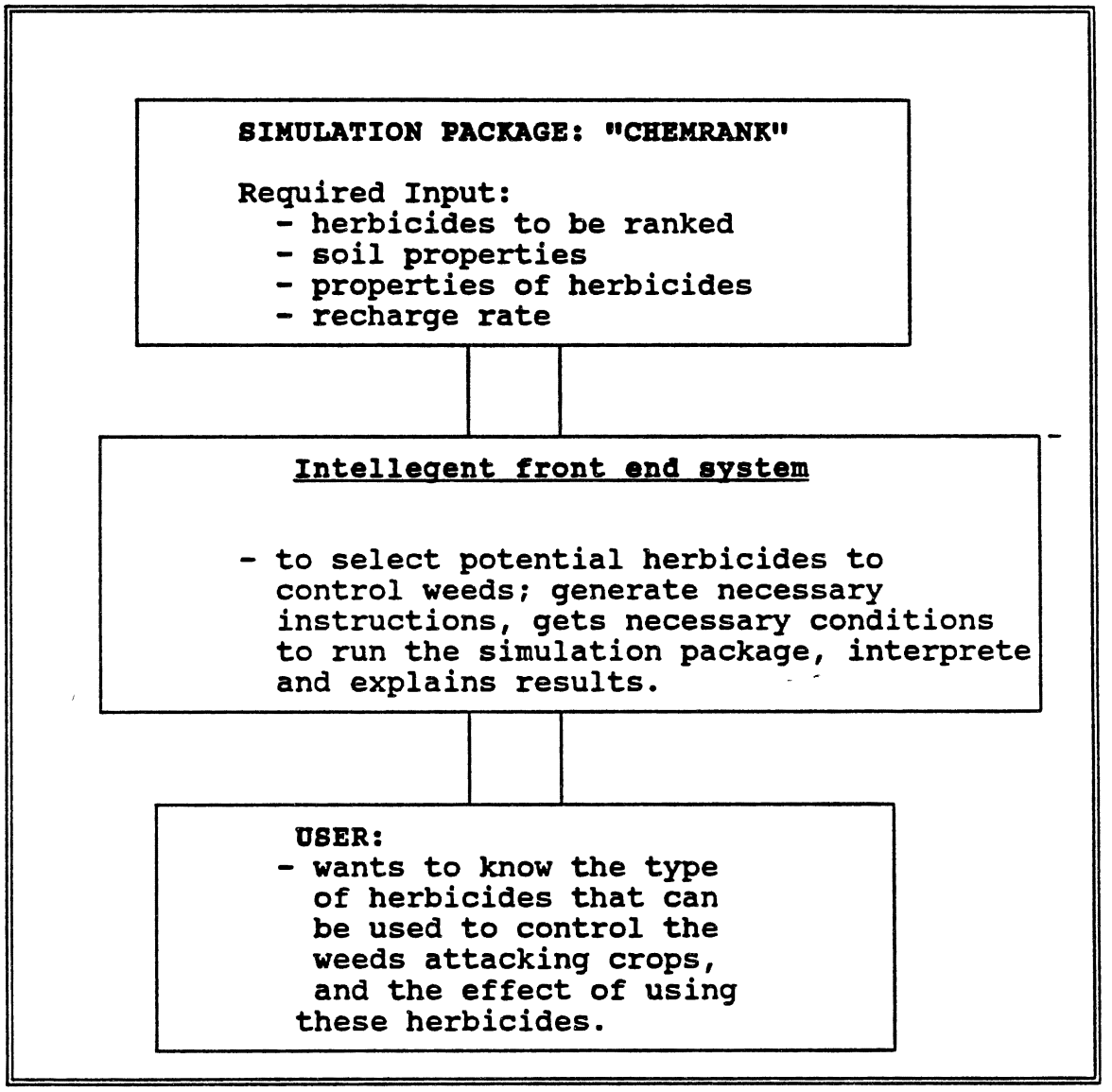

Figure.2.3. Intellegent front end system to select herbicides.

"CHEMRANK", and the user, generating necessary instructions, getting the necessary conditions to run the software through a dialogue with the user, and interprets and explains results obtained from the simulation package. The need to develop this expert system can be attributed to the following functions:

- static analysis: As the stated goals do not involve any time dependent information.
- mathematical analysis: several mathematical models are used to reach the goal (these models are explained in chapter IV).
- diagnostic analysis: The output of these models are used to determine the herbicides that can be used to control the weeds.

As described in Chapter IV, we use different mathematical models to simulate the rate at which herbicides would leach past the soil to affect the groundwater. Different mathematical models used for simulation are presented in section 4.3 of chapter IV. The simulation is carried out to compute the rank of the herbicides. These models namely the attenuation factor method and the retardation factor method have been used to simulate the rate at which herbicides would leach past the soil. The simulation model used is called "CHEMRANK" [8].
In the present work, this simulation package "CHEMRANK" is modified so that the necessary input for its running is obtained from the expert system.

Some of the input parameters required to run "CHEMRANK"

are the type of soil, soil conditions, name of herbicides, properties of herbicides, recharge rate and so on. Originally, these input parameters are obtained through an interactive menu-driven techniques, allowing the user to select these parameters. The simulation performs the analyses of these herbicides based on both the attenuation factor method and the retardation factor method.

However, in the present work the following modifications are carried out on this simulation package. First, changes are made to the interactive version of "CHEMRANK", so that the necessary conditions to run the software are obtained from the expert system through a dialogue with user, and based on several rules. For example, instead of asking the user the type of herbicides, these herbicides are selected by the expert system based on type of crop, type of weed, weed conditions, and the field conditions. These modifications and the different ways to get the necessary conditions to run the simulation package are explained in section 4.2 through 4.3 of chapter IV.

Second, instead of using both the mathematical models to rank herbicides, as in the original version of "CHEMRANK", only the attenuation factor method is used. the present work. The attenuation factor model is explained in the section 4.3 of chapter IV.

The attenuation factor method is used in computing the attenuation factor for each of the herbicides. The KBES, carries out the analyses to rank these herbicides. The KBES also interprets these results and present the user with the

results of the analyses. The KEBS computes the rank based on the fact that the larger the fraction of herbicides leaching past the specified soil depth, the greater the potential for the groundwater contamination. The analyses carried out by the KBES are explained in section 4.4 of chapter IV.

The knowledge base in the present work contains all the necessary control structure to select the potential herbicides for weed control in peanuts.

Although, the knowledge based simulation system presents lot of advantages in improving the analysis carried out, still there are some limitations, as described below.

2.10. Limitations of combining Expert Systems and Simulation

Some of the limitations of combining expert systems and simulations are given below.

- Interpreting knowledge about a system and it's performance is difficult. In other words, although a person is aware of the behavior of a particular system, he may not be able to interpret the same in the KBSS. Also, an expert's understanding about a system performance may change constantly and grow, as new situations are encountered. These new ideas have to be incorporated as knowledge into the knowledge based simulation system.
- It is easy to take the best expert systems available and assume that since it can solve complex problems, it

can obviously solve the lighter ones. But, this is not always true, since such expert systems often use a very complex way to solve a simple problem.

- The complexity of a knowledge base in most of the cases is directly proportional to the complexity involved in simulating a process. A user who intends to learn this new simulation will have to undergo tedious efforts to interpret the knowledge base also, which is in no way related to his field.

CHAPTER III

THE CxPERT SHELL

3.1. Introduction

In this chapter, the features of CxPERT shell are presented, including a description about the different ways of using the shell in the development of an expert system and different ways to represent domain knowledge. Finally, an evaluation of the CxPERT shell is presented.

3.2. Development Environment and Cycle

The CxPERT shell allows expert system developers to develop their expert system in a highly modular manner, while allowing procedural functions to be written in C and be invoked from the knowledge-base environment [6]. This is made possible by the CxPERT shell's property of being 100% compatible with C. The developer is also allowed to embed knowledge bases into existing C applications. Thus, the knowledge base of an expert system, could be encoded by using the Knowledge Representation Language (KRL) of the CxPERT shell, and/or the C language. Further, it is not necessary to encode the entire knowledge about the problem

domain into one knowledge base module. The knowledge can be divided into several knowledge base modules to suit the natural division of the knowledge. It is necessary for all these knowledge base files to have an extension of ".kb". These knowledge-base modules can be developed using any text editor.

These knowledge-base modules can then be translated into C source code using a CxPERT utility program for later compilation. Any syntactic errors in representing knowledge using the KRL would be enumerated at this stage and would have to be corrected. If the syntax check is successful, the knowledge bases are translated into C source files, compressed knowledge bases , and if required, header files.

These C files are then compiled by including the header files of all the knowledge base modules. It is possible that during compilation, the compiler may detect some additional errors not found by the syntax checker. If so, appropriate corrections have to be made in the knowledge-base modules.

The final step involves the linking of the object modules with CxPERT support libraries in order to get the executable version [6].

Executing the application program is done in exactly the same manner as any other C program. No additional software is required to run the executable version. However, during the execution, all the application's compressed knowledge base files (.kbc) must be available.

3.3. Knowledge Representation using CxPERT shell

The knowledge about the problem domain is represented using the CxPERT shell's Knowledge Representation Language (KRL) and C constructs. The KRL consists of attribute definitions, frame and slot definitions, advise statements, procedures and rules [6]. The KRL constructs used in the present work are briefly described in the following sub-sections.

3.3.1. Attributes

Attributes are one of the methods for representing knowledge, provided by CxPERT shell. It is a method of defining a characteristic of an object. An attribute's instantiation can be controlled as per the developer's requirement, that is, either through rules, procedures or advise statements. The instantiation of an attribute can lead to procedure calls, querying the user, or firing rules. An example of an attribute, whose syntax is typical of all attribute definitions, follows :

```
ATTR      integer height
ask       What is your height in inches ?
explain   In order to develop a complete profile of you.
values    all
endattr
```

ATTR : a keyword which identifies the start of an attribute definition.

integer : is one of the seven attribute types, the other types being long, float, double, logical, phrase and menu.

height : label of the attribute.

ask : a keyword which identifies the query to be displayed to the user.

explain : a keyword which identifies the explanation displayed to the user.

values : a keyword which identifies the values the attribute can hold.

endattr : a keyword which defines the end of attribute definition.

3.3.2. Procedures

CxPERT procedures correspond to C functions and are invoked in the same manner. They identify code which has been written in CxPERT KRL and which needs to be translated. Procedures allow the developer to break the code comprising the knowledge into several small parts which are easily referenced. The syntax of the PROCEDURE statement is as follows :

```
PROCEDURE name(argument,...)
argdcl,...
begin

body of the procedure

endprocedure(returnval)
```

PROCEDURE : a required keyword.

name : is the name of the procedure.

argument : is an optional C coded argument or arguments.

argdcl : are the C coded declarations for any defined arguments.

begin : a required keyword which must follow the PROCEDURE statement or any optional argument declarations.

body : of the procedure is the CxPERT KRL and C code that makes up the procedure.

endprocedure : a required keyword indicating the end of a procedure.

(returnval) : is an optional return value from the procedure.

CxPERT requires that there be a procedure named 'main', in every expert system developed using CxPERT.

3.3.3. Advise Statements

An advise statement is used to present to the user any textual description. The advise statement can either be directed to a hyperwindow (as explained in Sec.3.5), to a file, to a printer or to any combination of the above three. If the name of the hyperwindow is not given, then the advise statement directs the text to the default consultation window. It is also possible to direct the test to a particular location in the window. The values of C variables, CxPERT attributes and slots can be displayed using the advise statements. The syntax for the advise statement is as follows :

advise linenum "win_name" (filename,printflag)

Body of the advise statement

endadvise

advise : is a required keyword.

linenum : is an optional integer value which controls output to the screen.

win_name : The name of the window to which advise is to be directed. It is specific only to HyperWindows.

filename : Name of the file to which advise needs to be directed.

printflag: An integer (0 or 1) which indicates whether or not advise text is to be directed to the printer.

Body : The text to be directed.

endadvise: A required keyword indicating end of advise statement.

3.4. Rules

Knowledge about the problem domain can be coded as nested IF-then-ELSE rules. Rules can be used in procedures, attribute definitions, or slot definitions. CxPERT does not limit the content or the complexity of either the antecedent or the consequent of the rule. CxPERT supports an optional textual description of the rule. While processing a rule, CxPERT will process only as much of the antecedent as is necessary. The structure of a typical rule is given below:

```
rule
     textual description
endrule
IF cond then
     body of then
endif
ELSE
     body of else
endif
```

rule : Rules can be used in Procedures, Attributes, and Frames.

textual description : English descriptions of rules are supported to provide a more friendly end user environment.

endrule : A required keyword to end the textual description of the rule. It is used only if the keyword **rule** has been used.

IF : IF/endif are required keywords marking the

beginning and the end of the rules.

cond : Some conditional C expression which evaluates to true or false.

then : It is a required keyword.

body : It is a C statement/statements executed when the IF is true.

ELSE : ELSE is an optional keyword which has the same function as a C else.

body : It is a C statement/statements executed when the IF is false.

endif : It is a required keyword.

3.5. User Interface

CxPERT provides several interface options to the user. The most common among them are the hyperwindows. Hyperwindows are used to display textual information and/or to support hypertext. CxPERT has 9 default windows and additional windows can be created using a CxPERT window generation utility program. It is also possible to create the windows at run time. The user can also modify the window attributes (like size or color of the window) to his liking any time he wishes. The text directed to a window may be of any length and may contain references to CxPERT attributes or C variables. If the system detects that the user is not presented with the entire text, it automatically sends a message informing the user that additional text is pending. The user can thus view the remaining part of the text.

Several pre-defined functions of the CxPERT, such as

help, why, quit, and error functions can be invoked by the use of some special keys on the keyboard. The different special keys and the predefined functions they invoke are presented in APPENDIX-C. The developer can redefine these special keys to invoke other functions from the keyboard [6]. It also allows the developer to invoke any CxPERT procedure, any user-defined C function or any C library function. For example, a database or spreadsheet application may be integrated into the application.

In case the user gives an invalid answer to a question, the Expert System would display an error message and allow him/her to modify his/her response. The user can also quit the consultation any time he wishes.

3.6. Features of CxPERT system shell

The salient features of the CxPERT system shell are as follows :

a. Intermix of procedures written in C language;
b. Simple syntax at the shell level, which makes system development easier;
c. No prior knowledge of C language or syntax of the CxPERT knowledge representation language;
d. All the knowledge base, rules, procedures etc., written using CxPERT syntax are converted into the C language as explained in the previous section;
e. Since, the final code is compiled using either Microsoft C, Turbo C, or Lattice C, the executable

version which is created is portable on any of the IBM compatibles;

(test runs are made on IBM compatibles).

f. Easy way to create hyperwindows and hypertexts.

g. Hiding sensitive rules so that they do not form a part of an explanation given to a user when the user wants to know why a particular question is being asked, or how a particular state has been reached.

The CxPERT development cycle is depicted in Figure.3.1.

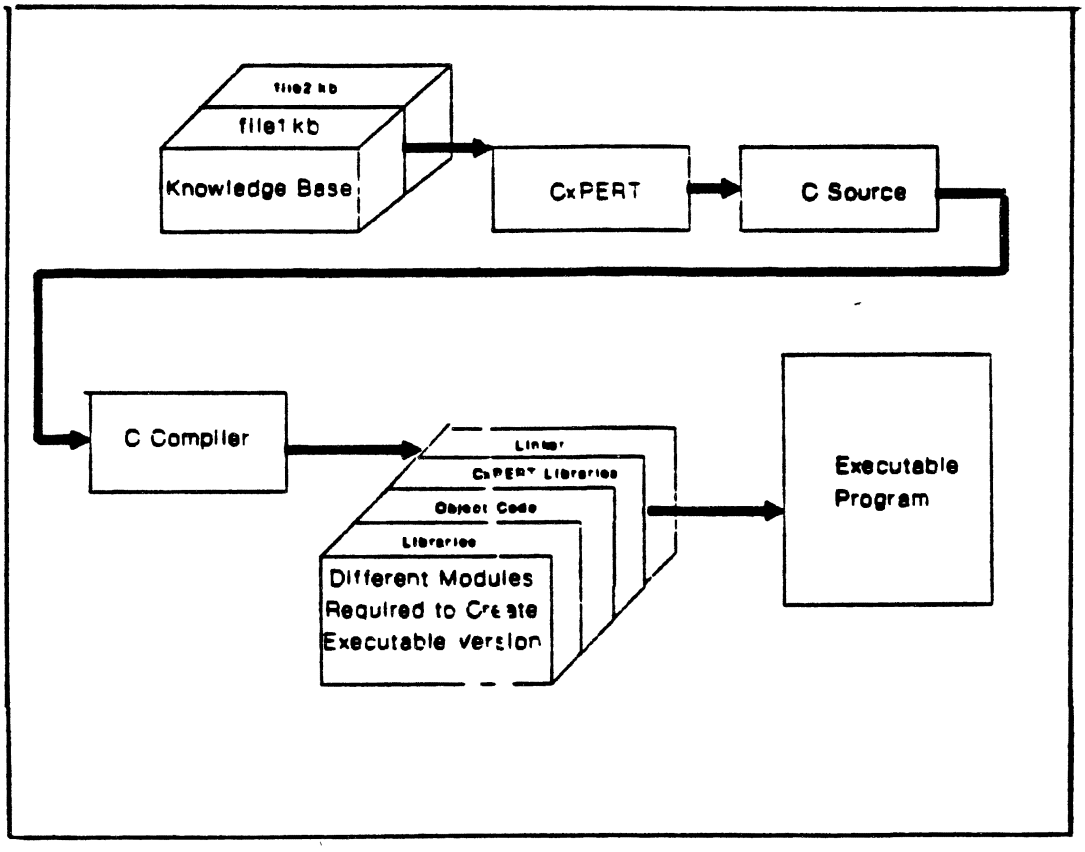

Figure.3.1. CxPERT Development cycle.

CHAPTER IV

DEVELOPMENT OF THE EXPERT SYSTEM

4.1. Introduction

The aim of this work was to develop an expert system that would interface with a simulation package and help the user to select herbicides to control weeds attacking the user's crops, based on economical and environmental considerations. The major design issues which were considered while developing this hybrid expert system were maintainability, expandability, and portability.

Thus, it was decided to keep a majority of knowledge in the form of facts about the problem domain separate from the knowledge base (consisting of rules and some facts).

For the purpose of a shell, CxPERT was decided upon, since it has an easy interface with C language, thereby making the development of the expert system easier.

Following decisions were made in the development process:

- Since the crop data file was quite small, a sequential search for the user specified weed and field condition in the specified crop's data file was deemed sufficient. The crop data file

consists of crop name, weed number, weed name, the type of herbicide that can be used to control the weed, field condition, and the percentage control.

- If the user decides to update the cost database, since total number of herbicides whose cost to be updated can vary for different weeds and field conditions, a linear linklist is used to store the information. The cost data base consists of two fields. They are the weed number and the cost per gallon of the herbicide.
- Since the soil data file was large, an index file for this soil data file with the soil index and the byte offset as two fields is created. This is designed in such a way, that each time a user introduces a new soil, the index file has to be updated using a separate module written in C language.
- A binary search on the index file is carried out, based on the soil identifier. The corresponding byte offset is used to move the file pointer and read the soil properties.
- The cost computation for the combinational herbicides are carried out separately. This is necessary, because, at the time of development of this expert system, rules are not available to compute the cost of the combinational herbicide. Computing the cost separately makes

the future development easier.

The Expert System development consisted of three phases which are described in the following sections.

4.2. PHASE - 1 of the Expert System development

PHASE - 1 of the expert system development, as depicted in figure 4.1, consists of the following four functions :

* Knowledge acquisition;
* update the Cost Database;
* Get the Weed name;
* Select the potential herbicides.

The knowledge acquisition for the present work is depicted in the figure 4.2. As explained in the figure, rules are obtained from experts in this field [14]. The knowledge base is developed from these rules.

The information about the type of crop, weed name, and the soil texture is obtained from the user through the queries. The weed names are presented to the user as a menu, where the user selects the appropriate weed by entering the number corresponding to the weed name attacking his/her crop. A typical menu screen for the weed selection by the user is given in figure 4.3.

Next, the user is given an opportunity to update the cost database. If the user does not want to change the cost database, he can respond to the system by saying "NO". However, if the user wants to update the cost data base, then the possible herbicides for the control of the weed as

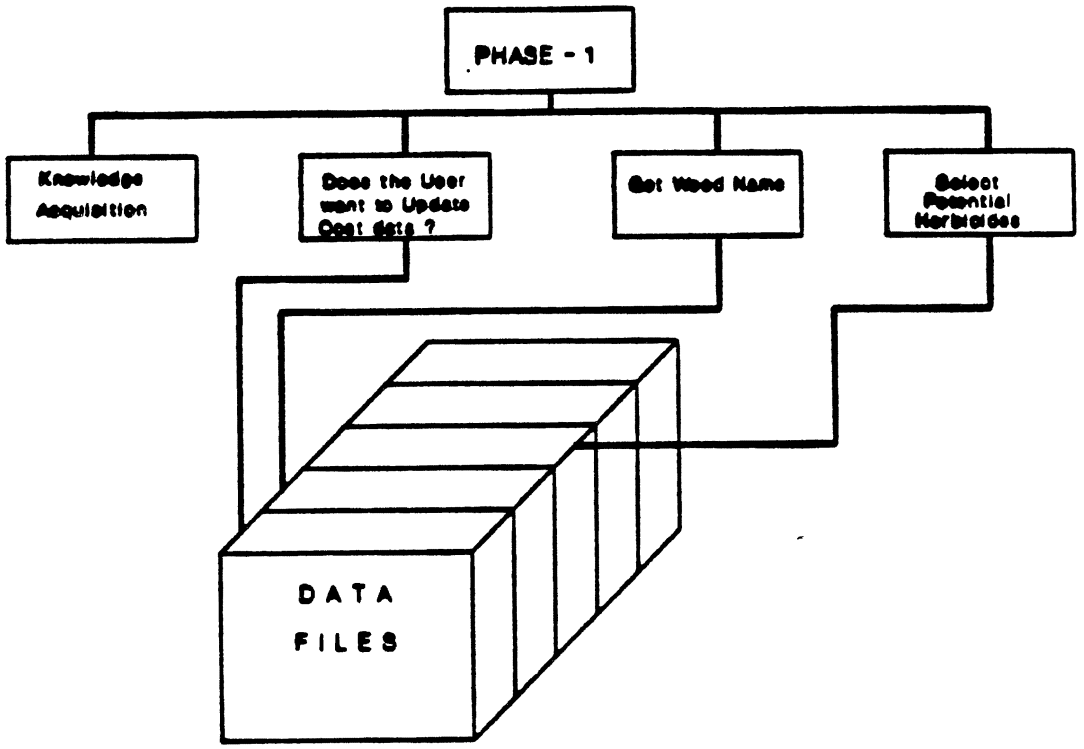

Figure.4.1. Functions of PHASE-1.

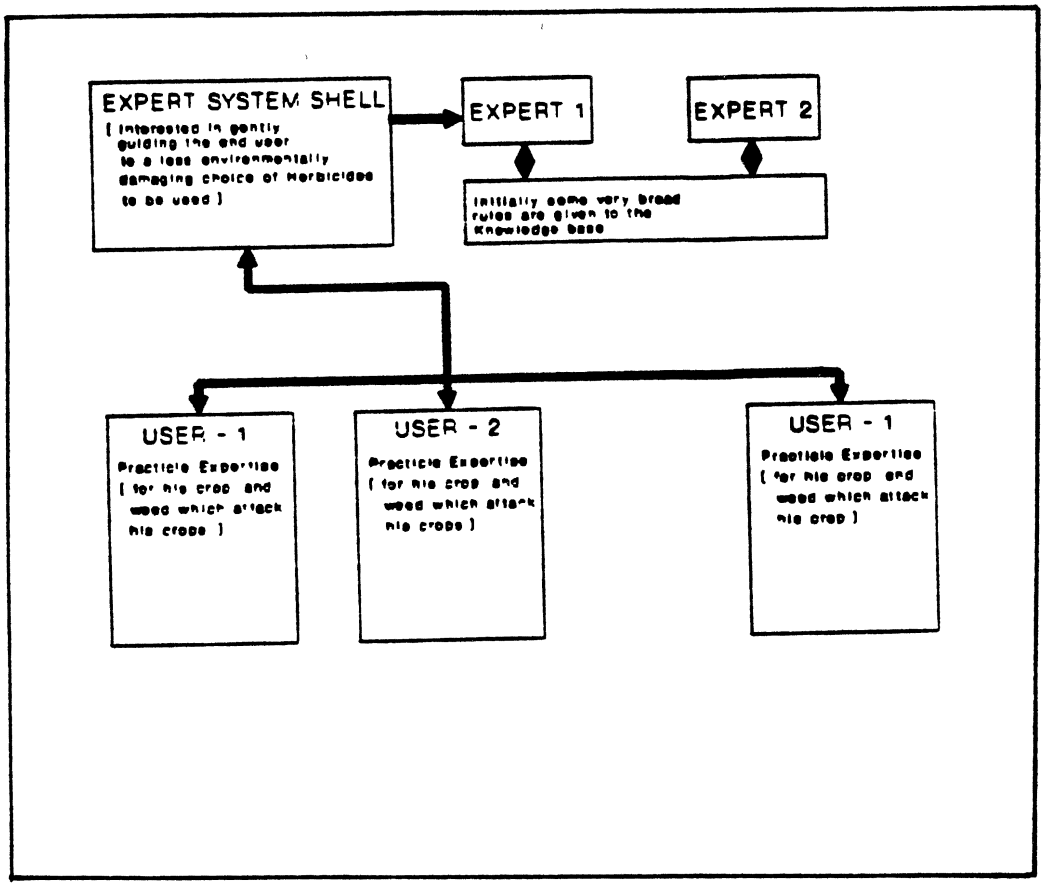

Figure.4.2. Representation of Knowledge Acquisition from different sources.

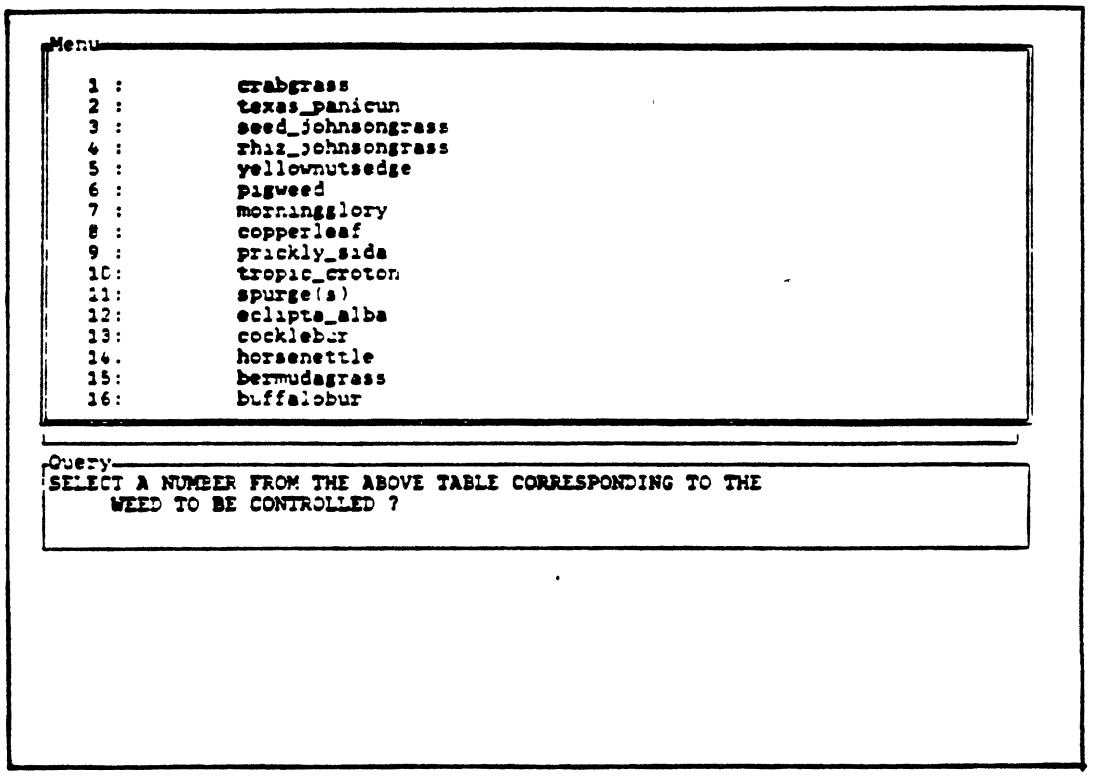

Figure.4.3. A typical menu screen for weed selection.

specified by the user, are first selected. This is done in order to allow the cost updating of only those herbicides, which are selected based upon the weed and the field condition specified by the user.

After getting the information from the user about the changes to be made in the cost database, the system displays several queries to get information about the crop, type of weed, field conditions and so on. Based on the weed name, the weed number is obtained from a data file. The program is implemented keeping in mind the possibility of introducing new weeds. Suppose, new weeds are to be introduced, this can be achieved just by adding the name of the weed in the knowledge base in which the weed's attributes are defined. All the weeds are presented to the user as a menu, from which the user is asked to select the weed attacking his crop. Based on the weed, the selection of different herbicides is done by reading the herbicide database. The herbicide cost database consists of weed names, herbicides that can be used to control the weeds, the percentage of control, and the type of application desired. Different herbicides can be used to control the same weed. Depending upon the choice of weed, the program selects all those herbicides that can be used to control that weed. Since certain herbicides can be used only in particular field conditions, field condition is used as one of the parameters to determine the correct herbicide(s).

The field condition value is set based on the following conditions:

a. If the crop is not emerged then the field condition value is set to 1.
b. If the crop is planted and weeds are not emerged, then the field condition value is set to 2.
c. If the crops and weeds are emerged, and weed age is less than 6 days then the field condition value is set to 3.
d. If the crops and weeds are emerged, and the weed age is greater than 6 and less than 13, then the field condition value is set to 4.
e. If the crops and weeds are emerged, and the weed age is greater than 13, then the field condition value is set to 5.

The flow chart for the selection of the herbicides based on the type of crop and the field condition is given in figure.4.4. Initially, the information about the crop, weed, and soil texture is obtained from the user through queries. The field condition value is set based on the crop emergence, weed emergence, and the age of the weeds. After setting the field condition, the soil texture is obtained from the user through the menu. The soil texture is required in order to compute the herbicide rate.

If there are no herbicides corresponding to the weed name and the field condition specified, the system displays a message to the user about the non-availability of the herbicides. This gives the user the option either terminating the session or changing the field condition.

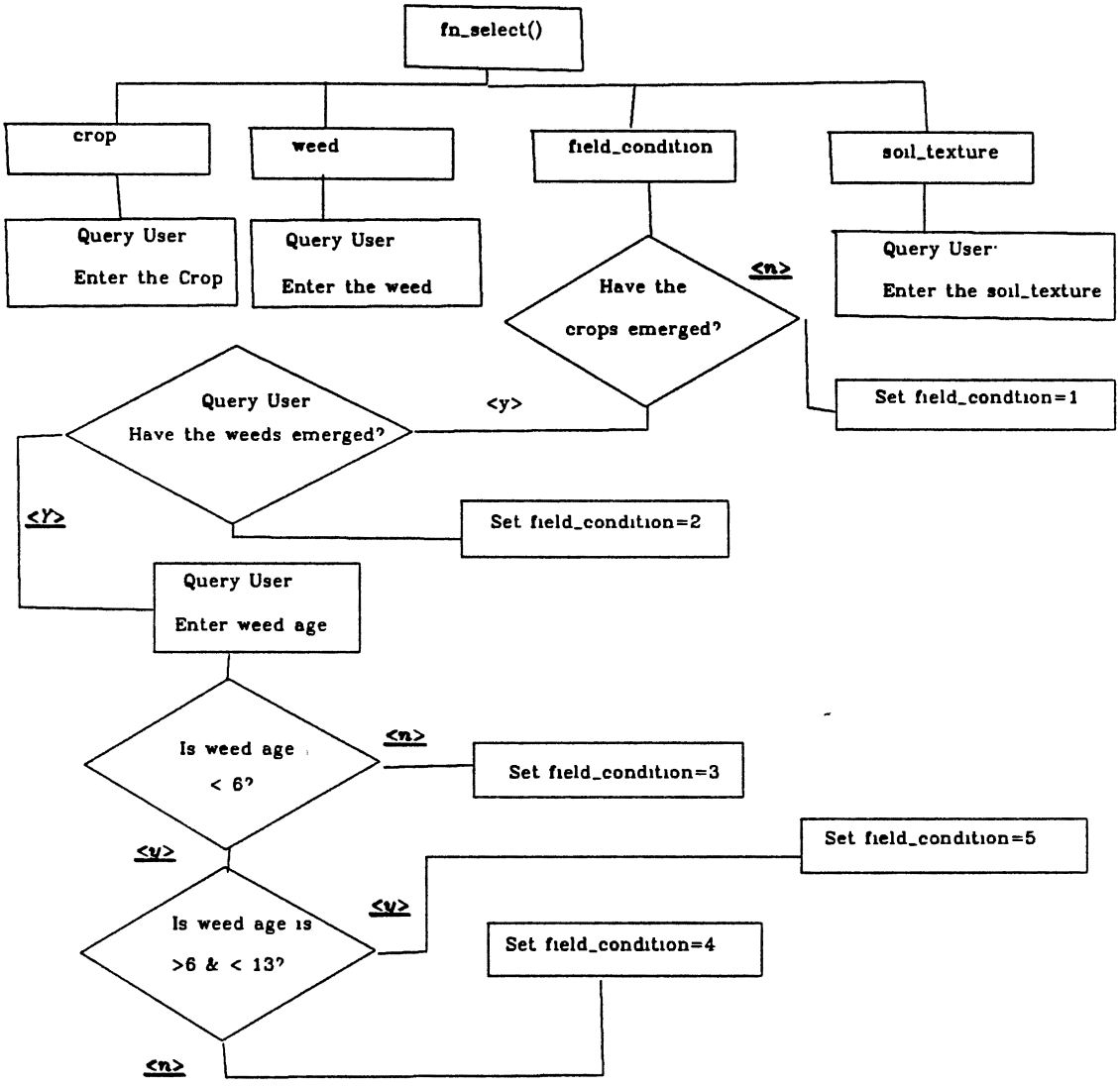

Figure.4.4. Flowchart for function fn_select().

The modules for selection of the herbicides are written in C language. Instead of having the herbicides as part of the knowledge base, a separate modular C code is developed. This is done in order to make the expert system more general in handling different types of crops. By replacing the current database by a database of different crops, it is possible to run the system without any further change. However, if the knowledge about all the herbicides corresponding to a particular crop had been embedded in the knowledge base and if each time a new crop had to be analyzed for the control of weeds, it would be necessary to embed the knowledge about the crop. This would result in making the system more complex and the generality obtained would be lost.

Variables for the stress conditions are set based on the information obtained from the user based on the weed conditions such as, if the weeds are actively growing, or if the weeds are stressed etc. Stress conditions variables are required, in order to enable the program to decide if a second application of herbicides is required to control some of the weeds. For example, stress conditions (hot,dry) will reduce the effectiveness of broadleaf herbicides Basagran, Blazer, and Storm. If grasses present are undergoing severe moisture stress, then post emergence herbicide Poast should not be used. Also, if the weeds are emerged for more than 13 days, weed control may be reduced even with the suggested herbicides and a repeat application may be required.

4.3. PHASE - 2 of the Expert System development

After selecting the herbicides, the soil name is obtained from the user through the menu. Since it is difficult for the user to select soil names correctly, it was decided to present the user with all the soil names and then allow the user to select the soil from the menu. A typical menu screen for the soil identifier is given in figure 4.5. Based on the soil selected, properties pertaining to the soil are obtained. In PHASE - 2, the System initiates the Simulation program in order to rank the herbicides selected by PHASE - 1 based on their effect on ground water. This is done by using a simulation package called "CHEMRANK" [8]. Two schemes are supported by this software to rank the chemicals [8]. They are:

* Retardation Factor (RF)
* Attenuation Factor (AF)

Presently, only the Attenuation Factor method is used to simulate the ranking of the selected chemicals. Attenuation Factor (AF) is an index which gives the fraction of the applied amount of chemical that is likely to leach past a specified soil depth. It is assumed that the larger the fraction of chemical leaching past the specified depth, the greater is the potential for groundwater contamination [8]. The relative amount of chemical passing through the soil in time t_1 is given by

$$M_1 / M_2 = \exp(-k_1 * t_1) \qquad \text{----------[4.2a]}$$

where M_0 and M_1 are the mass of chemical at the top and

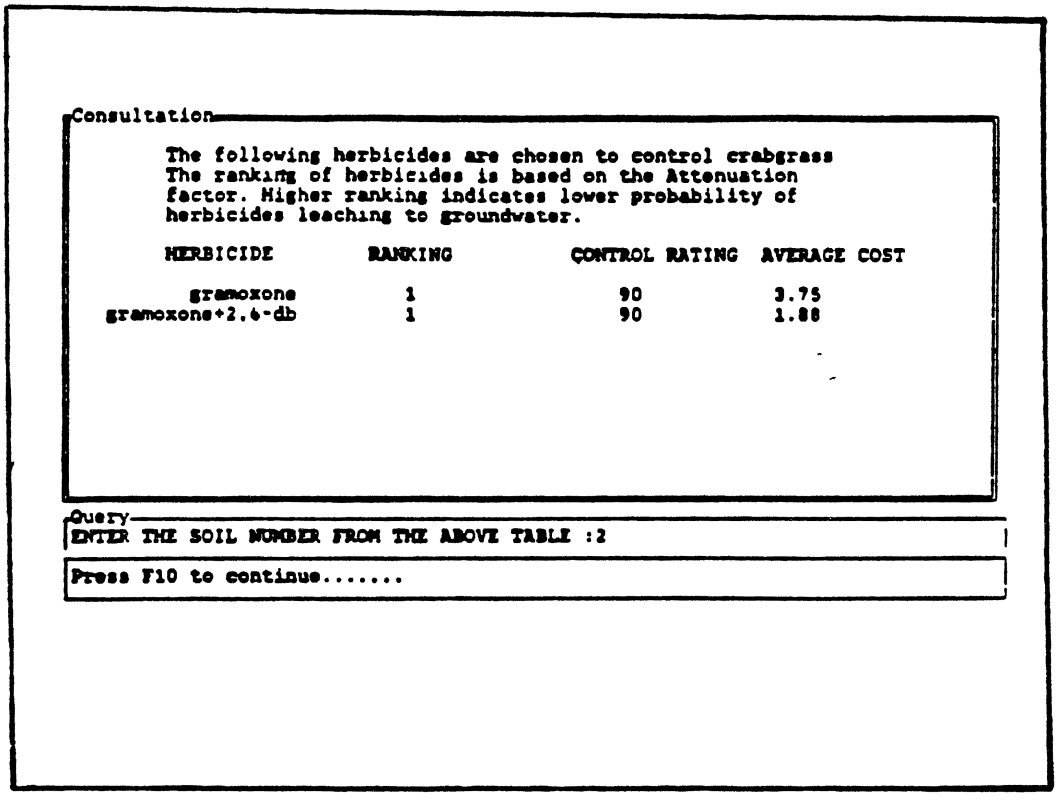

Figure.4.5. A typical menu screen showing ranks.

bottom of the soil, respectively, and k_1 is the degradation constant. In this simple case the attenuation factor and degradation constant are respectively given

$$AF = \exp(-k_1 * t_1) \qquad \text{----------[4.2b]}$$

$$k_i = 0.693 / \text{half-life}_i$$

where half-life_i is the half life of the chemical in layer i.

For a soil with multiple layers having different degradation rates or travel times, equation 4.2a can be applied to each layer. If M_N represents the mass moving out of the bottom layer N, then

$$M_N / M_0 = \exp(-k_1*t_1-k_2*t_2-.....-k_N*t_N) \qquad \text{------[4.2c]}$$

and the attenuation factor is given by

$$AF = \exp(-k_1*t_1-k_2*t_2-....-k_N*t_N) \qquad \text{------[4.2d]}$$

Equation 4.2d forms the basis for calculating the attenuation factor for layered soils. For a layered soil, the time required to move through N layers to the depth of interest is the sum of the times required for each layer. If d_i, i=1,2,3,...,N, represents the depth of the bottom of each layer and $d_0 = 0$, then the time, t_i, required to move through layer i is given by

$$t_i = (d_i - d_{i-1}) * RF_i * O_{FCi} / q_i \qquad \text{--------[4.2e]}$$

where O_{FCi} is the field capacity for the layer i, q_i is the recharge rate through layer i, and RF_i is the retardation factor for layer i. The recharge rate is assumed to be constant for all the layers. RF_i is given by the following equation.

$$RF = 1 + (rho*K_d + (f - O_{FC})*K_h)/O_{FC} \qquad \text{---------[4.2f]}$$

where rho is the bulk density of the soil, f is the porosity of the soil, O_{FC} is the volumetric soil-water content at the "field capacity", K_D is the partition coefficient for the chemical in the soil, and K_h is the dimensionless Henry's constant for the chemical [8].

Depending on the Attenuation factor (AF) ranking of these herbicides is carried out. In ranking the selected chemicals it is assumed that larger the fraction of herbicides leaching past the specified depth, the greater is the potential for the groundwater contamination.

4.4. PHASE - 3 OF the Expert System development

In PHASE - 3, the system assigns rank to each of the herbicides selected, based on the value of the attenuation factor. It displays to the user the ranking of the herbicides, the quantity of herbicides to be used, and the total cost of the herbicides. Also, if the user wants a rerun, the attributes presented by the user is displayed, allowing him/her to make changes.

Finally, the system enquires if the user is interested in either storing the results of that particular session in a file or to have a hardcopy of the results.

4.5. Implementation of PHASE - 1, PHASE - 2, and PHASE - 3

The implementation of PHASE - 1 was carried out as

follows:

Initially, the database containing the crop name, weed number, weed name, type of herbicides, field condition, and the percentage of control rating was created. In the present work, the system was tested only with the data for the crop 'peanuts'. In order to maintain the readability of the database, a fixed length record with fixed length field was decided upon.

After the database was created, the knowledge acquired from the experts[7,14] was represented using CxPERT shell syntax. In order to select the weed name, a menu was presented to the user as shown in the figure 4.3. The information necessary to select different herbicides is obtained by the user, through some rules which are fired according to the user response. Based on the information provided by the user, such as weed name, field condition, and soil texture, herbicides are selected. The procedure for selecting herbicides are written in C language. Testing of PHASE - 1 was carried out as follows.

First several runs were made to identify the problems incurred during the development process. For example, when the field condition is 5, only a few herbicides are available. That is, herbicides may not be available for certain weeds under certain field conditions. One solution for this problem is to terminate the session, informing the user that no herbicides are available to control the weed for the given field condition. A second and more appropriate solution is to ask the user to change certain

conditions. Also, if the user is not able to decide about the field condition, then certain rules are developed to set the field condition which are explained in the section 4.1 of this chapter.

PHASE - 2 consists of modifying the simulation package "CHEMRANK," developed in C by Nofziger et al. [8], to suit the present work's requirement. In CHEMRANK, it is required that the user selects the ranking schemes of interest, specify the herbicides to be included for ranking and their properties, soil and several other properties [8]. These are carried out in CHEMRANK through the use of menus. However, in the present work, the selection of herbicides, the properties of herbicides, selection of the soil and their properties are selected by the expert system through the use of several queries and rules. Changes are made in CHEMRANK so as use the parameters obtained by PHASE - 1. The following modifications are carried out on CHEMRANK:

-In order to select the soil, different soils are presented to the user as a menu, and the user is asked to select the soil identifier corresponding to the soil. Based on the soil selected by the user, the soil properties are read from the soil database. The soil database developed by Nofziger et. al. [8] is retained.

-From the PHASE - 1, selected herbicides are passed to CHEMRANK to compute their ranks.

-Though CHEMRANK computes the ranks of the herbicides based on all the methods such as the attenuation

factor method, retardation factor method, travel time from CMLS and so on, the simulation is reduced to only the Attenuation factor method in the present work.

PHASE - 3 is essentially a post processing operation. This is implemented using CxPERT shell syntax. This involves displaying the results such as ranking of herbicides, their control rating, cost of individual herbicides, and the total cost of combinational herbicides. Queries are also made to check if the user wants to have a rerun with changes in any of the parameters. The program also allows the user to store the results or to make a hard copy of the results.

4.6. Operation of the Expert System

The flowchart for the operation of the expert system is given in figure 4.6.

The expert system is started by querying, if the user wants to select the herbicides and rank them, or if the user wants to select the herbicides, update the cost database, and rank the herbicides. If the user decides for the later one, then the herbicides are selected, based on certain input conditions, as explained in section 4.1 of Chapter IV.

The selected herbicides are presented to the user along with their present cost. The user is asked to select the herbicide for which the change in the cost is to be made. A second chance is given to the user to confirm the changes made to the cost of the herbicide. After getting the

approval from the user the cost database is updated. Otherwise the original cost database is retained.

The selected herbicides are passed to the ranking package ("CHEMRANK") along with their properties. Several queries are made to gather information about the soil, their properties etc. After ranking the herbicides, a display of rank of herbicides, control rating, and the cost of herbicides are displayed to the user as shown in figure 4.8.

Queries are made to the user to determine whether to rerun or end the session. If the user wants a rerun, then the parameters which are used to select the herbicides, and the parameters which are used to rank the herbicides are presented to the user. Queries are made, which allows the user to change any of these parameters.

Before ending the session, query is made to the user to see if a hardcopy of the results is required.

A typical session of the operation of the expert system is given in APPENDIX A.

The hardware requirement for this expert system is given in APPENDIX B.

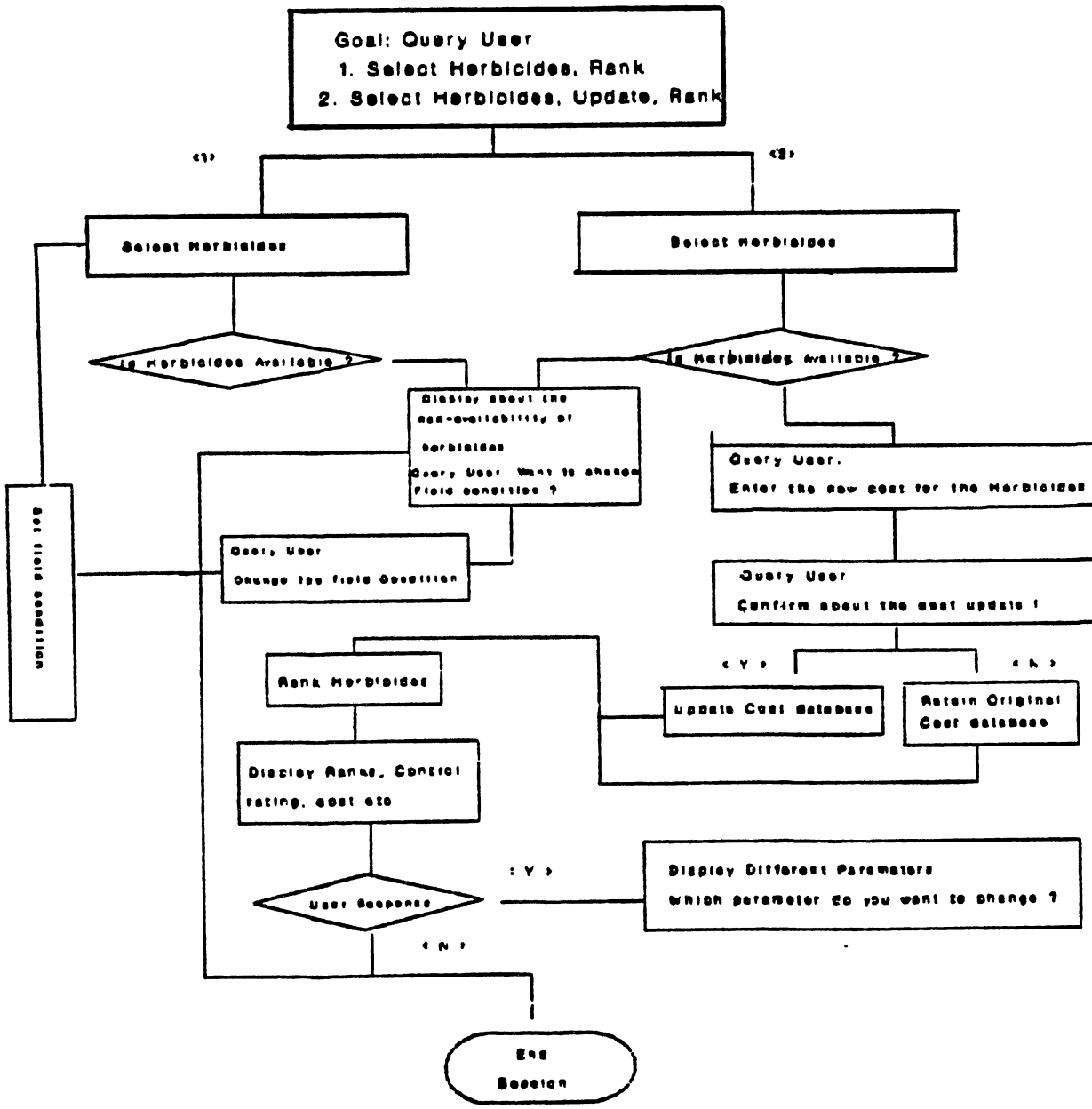

Figure.4.6. Operation of Expert system.

CHAPTER V

RESULTS

5.1. Introduction

This chapter lists out the conclusions made about the working of the expert system. Also, an explanation about the future developments those can be carried out, are listed.

5.2. Conclusion and Scope for future work

Based on several test runs with human experts [7,14], the following conclusions are made:

- Use of CxPERT shell is definitely a positive step in bringing together the power of Artificial Intelligence and the flexibility of the C language.
- Combining the techniques of Simulation and Artificial Intelligence, results in a more realistic approach to the problems.
- Test runs were made on different IBM compatibles; it was observed that, by having an executable version along with the CxPERT shell

file "CxPROC.KBC", it is possible to execute the program.

- The expert system gave a good estimate for the ranking of the herbicides.
- Since the knowledge about the different crops was read from a file, the same expert system can be used for different crops. So, it is justifiable to have the knowledge about each crop in a separate database. But, care should be taken to have a similar format of all data bases.

5.3. Scope for the future work

- The present work was aimed at developing an expert system to handle a single weed. The same system can be developed to handle multiple weed problems.
- In many conditions, for an identified field condition, herbicides are not available. Since field condition is one of the main parameters used to select herbicide, some more analysis in the selection of field condition will help in the improved analysis.
- Many times, when different herbicides are selected, the rank and the control rating assigned to the herbicides by the expert system remains the same. So the user is given an option

to select one of the herbicides. Use of other methods like retardation factor, travel time from CMLS to assign rank to herbicides will help in narrowing this to a particular herbicides. In other words, some more rules can be used to assign the ranks.

- In case of multiple herbicides, it is assumed that all the herbicides will be used in the same quantity. Instead, one can think of a different rule to handle the above stated condition.
- Only one ranking scheme based on the attenuation factor is considered. Other techniques to rank herbicides such as, retardation factor, travel time from CMLS should be considered.
- A separate system level program would help the expert system to adapt to the different architecture.
- Only combination of two herbicides are considered. Combinational herbicides of more than two should also be considered.
- The confidence level of the analysis carried out by the expert system can be increased by taking into consideration the psychological factors involved in interfacing with the end user. One of the factors that could be considered is to consider the possibility, that the end user have some herbicides in mind to control the weed attacking the crop. Analysis can be carried out

on the herbicide/(s) presented by the end user. Later a comparison can be made between the herbicides selected by the expert system and the herbicides presented by the user.

REFERENCES

1. Addis. T.R., "**Designing Knowledge Based Systems**", Prentice-Hall, Englewood Cliffs, (1988) New Jersey 07632.

2. A.Dale Whittaker, Ronald H. Thieme, "**AI Techniques in Agriculture**", American Association for Artificial Intelligence, San Antonio, August 1988.

3. Ben Shneiderman, "**Designing the User Interface**", Strategies for Effective Human-Computer Interaction, Addison-Wesley, 1987.

4. Berg. C.H, Engelman. C, Bischoff. M, KNOBS:"**An Experimental Knowledge based Tactical Air Mission Planning System and a Rule Based Aircraft Simulation facility**". Proceedings, International joint conference on AI, 856-861 (1981).

5. Chris Naylor, "**Build your Own Expert System**", Sigma Publishers, Wilmslow, NY.

6. **"CxPERT USER'S GUIDE"**, Software Plus, Ltd., 1652 Albermarle Drive, Crofton, MD 21114, 1989.

7. D.L. Nofziger, Professor, Agronomy, Oklahoma State University, Stillwater.

8. D.L. Nofziger, P.S.C. Rao, and A.G. Hornsby, **"Interactive Software for Ranking the Potential of Organic Chemicals to Contaminate Groundwater"**, Agricultural Experiment Station, Division of Agriculture, Oklahoma State University, Aug.1988.

9. Eugene Charniak, Drew McDermott,"**Introduction to Artificial Intelligence**", Addison-Wesley publishing Co., 1986, Ca.

10. Herbert Schildt, "**The Complete Reference**", McGraw-Hill, Berkley, Ca.

11. **Herbert Maisel, Guiliano Gnugnoli, "Simulation of Discrete Stochastic Systems"**, Science Research Associates, Inc., 1972

12. James Martin, Steven Oxman, "**Building Expert Systems**",

A Tutorial, Prentice-Hall, Englewood Cliffs, New Jersey (1988) 07632.

13. James F. Clark, Judith J.Lambrecht, **"Information Processing concepts, principles, and procedures"**, South Western publishing Co., Palo Alto, CA.

14. Mayfield B.M, Assistant Professor, Computing and Information Sciences, Oklahoma State University, Stillwater.

15. Masanobu Watanabe, Toru Yamanouchi, Masahiko Iwamto, Yuriko Ushoda. CL:**"Flexible and Efficient Tool for constructing Knowledge-Based Expert Systems."**, IEEE Expert, Fall 1989, 41-50.

16. Lambrecht, Clark,**"Information Processing Concepts, principles, and Procedures"**, South-Weastern publishing Co., palo Alto, Ca.

17. Robert Keller, **"Expert System Technology, Development & Application"**, Yourdon Press, A prentice-Hall Company, Englewood Cliff, NJ 07632, 1987.

18. Peter S. Sell, **"Expert Systems- A practical Introduction"**, John Wiley & Sons, New York, 1986.

19. Pritsker, A. A. B., **"Compilation of definitions of Simulation"**, SIMULATION, Vol.33, 1979, pp.61-63.

20. Palanisamy Sivasubramaniam, **"IAT: An Intelligent ADA Tutor"**, M.S. Thesis report, Oklahoma State University, 1990.

21. Rajeev Sangal, **"What is Artificial Intelligence ?"**, Department of Computer Science and Engineering, Indian Institute of Technology, India, 1986.

22. Pearl, **"J., Heuristics: Intelligent Search Strategies for computing and problem solving"**, Addison-Wesley, 1984.

23. Rajeev Sangal, **"Computational Paradigm in Natural Language"**, Proc. of National Seminar on Artificial Intelligence, Indian Institute of Science, India, 1986.

24. Robert R. Mitchell, **"Expert Systems and Air-Combat Simulation"**, AI Expert, September 1989.

25. Richard Shannon, E., **"System Simulation: The art and science"**, Prentice-Hall, 1975.

26. Schrage, L.E, B.L.Fox, and Bratley, P., **"A Guide to Simulation"**, Springer-Verlag, 1983.

27. Waterman, D.A. "**A Guide to Expert Systems**", Addison Wesley Publishing Company, Reading, Massachusetts (1986).

28. Willanueva.R, Kiviat, P. J., and H.Markowitz, "**The SIMSCRIPT II Programming Language**", Prentice-Hall, 1969.

29. Willanueva R. Hill, and Stephen D. Roberts, "**A Prototype Knowledge-Based Simulation support System**", Simulation 48:4, pp.152-161, April 1987.

30. Weimo. H, Robert E. Shannon and Richard Mayer, Adelsberger, "**Expert systems and Simulation**", Simulation 44:6, pp.275-284, July 1985.

31. Waines, B. R. and Shaw, M. L. C., "**Expert Systems and Simulation**", AI Expert, pp.19-28, April 1988.

32. Vamana Reddy,"**Epistemology of Knowledge Based Simulation**", Simulation 48:4, pp.162-166, April 1987.

APPENDIXES

APPENDIX A

A SESSION WITH EXPERT SYSTEM

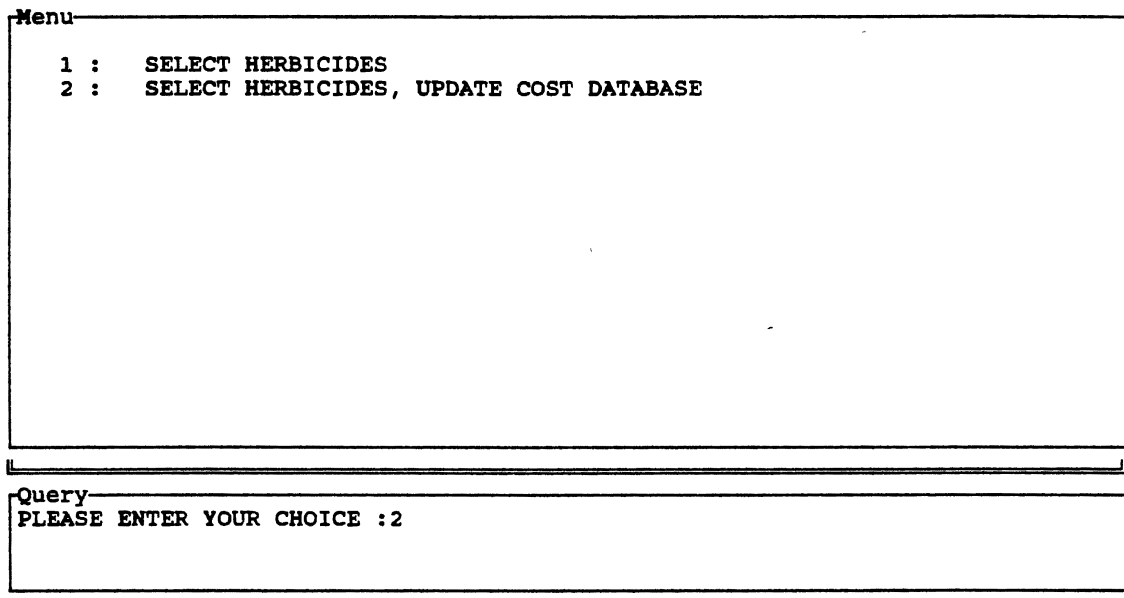

Figure a2. Query is made to the user, if the user wants to update the cost database and then select the herbicides, the user has to enter 2, else if only selection of herbicides are required, the user has to enter 1.

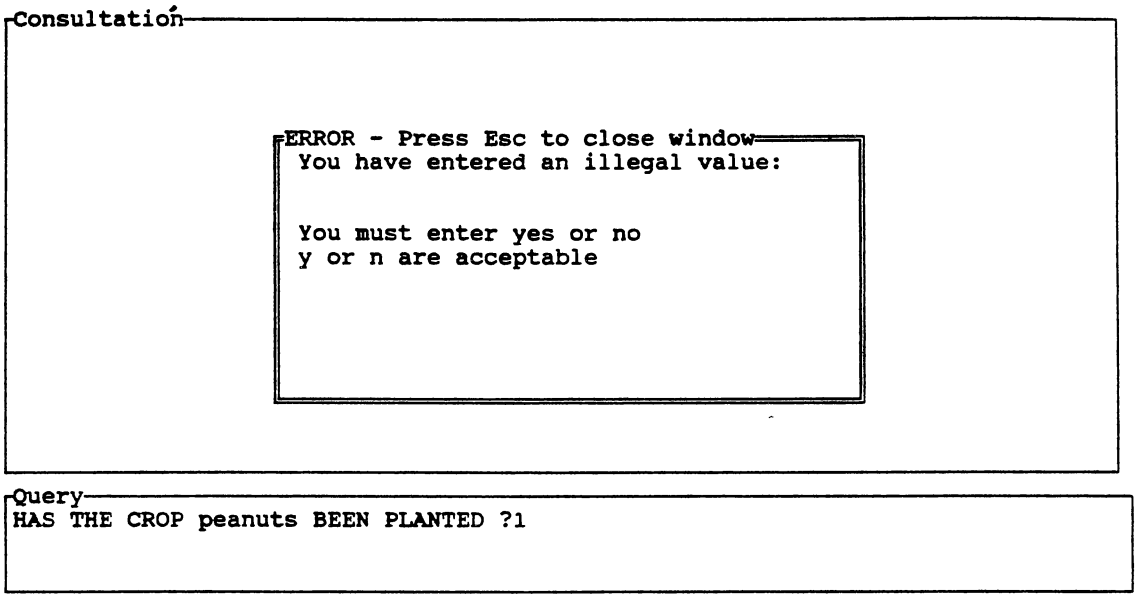

Figure a3. Query is made about the crop. This is required to fix the field condition. If the user enter an illegal value, as shown in the consultation window, then the system automatically informs the user about the legal values to be used. This is handled by the error function.

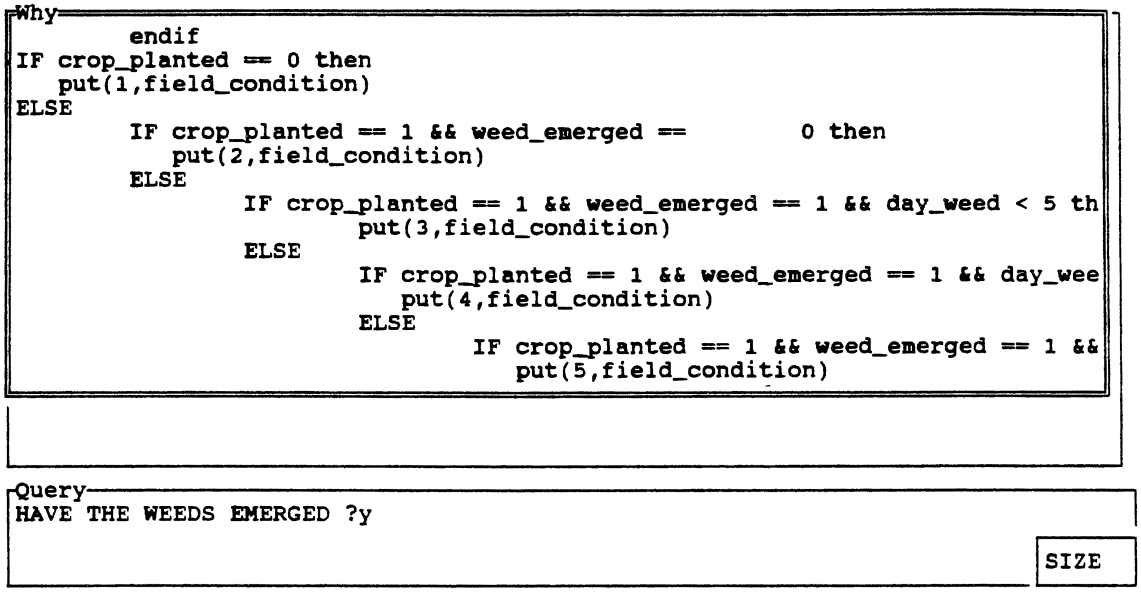

Figure a4. The user can get an explanation about why a query is being made or a decision is being taken. This can be obtained by pressing the default key F3 as shown in figure.

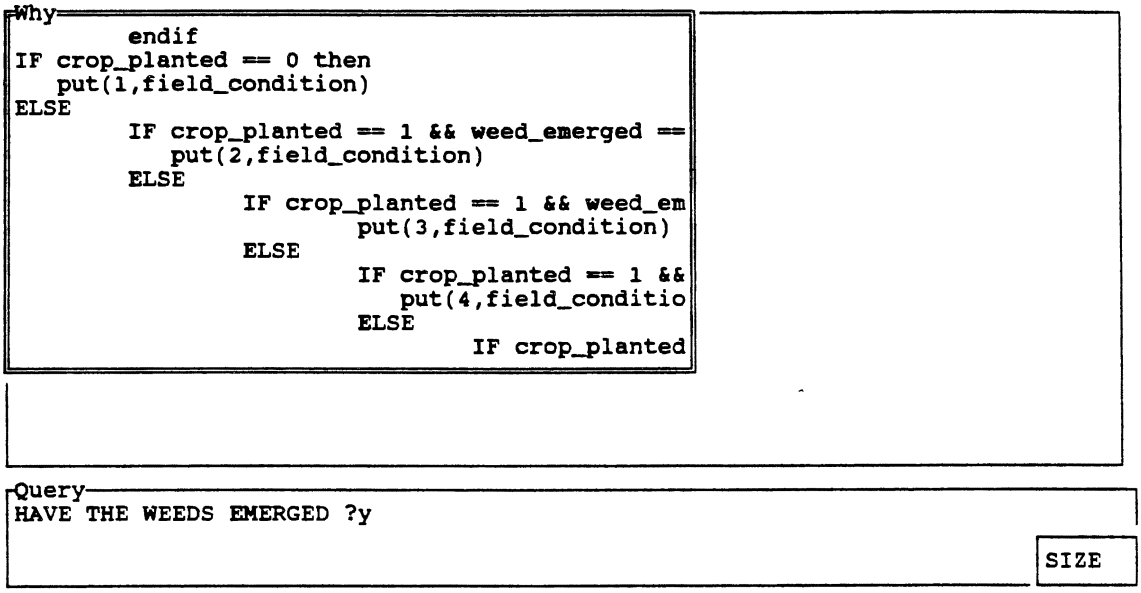

Figure a5. This figure explains how the size of a window can be changed. This can be achieved by using the default key F5. When this key is pressed, it gives the type of operation that can be performed on a window, like scrolling the contents of a window, increasing or decreasing the size of a window, moving a window and so on.

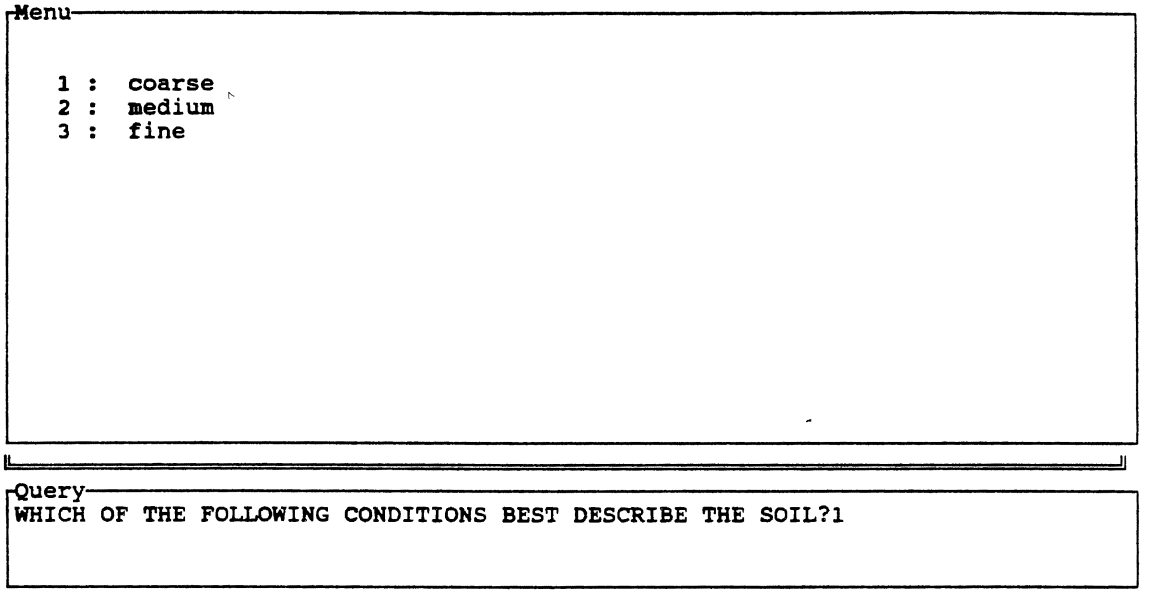

Figure a6. This figure explains how a user can choose the type of the soil in his fields.

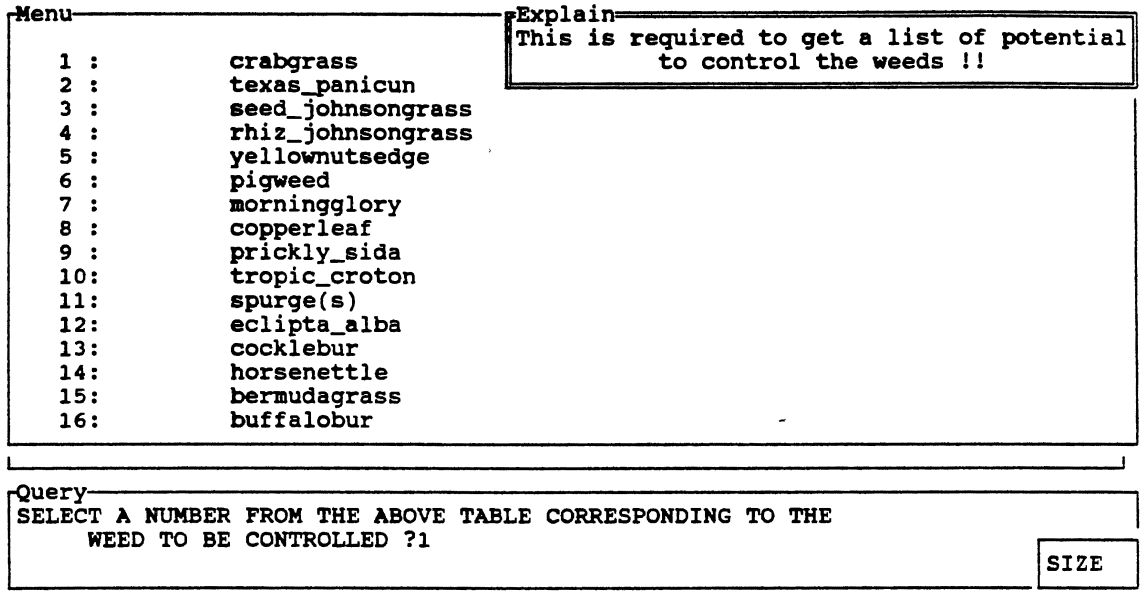

Figure a7. This figure explains how a user can choose the type of weed attacking his crops. It also shows how to get an explanation about a particular query. This can be obtained by the default key F1.

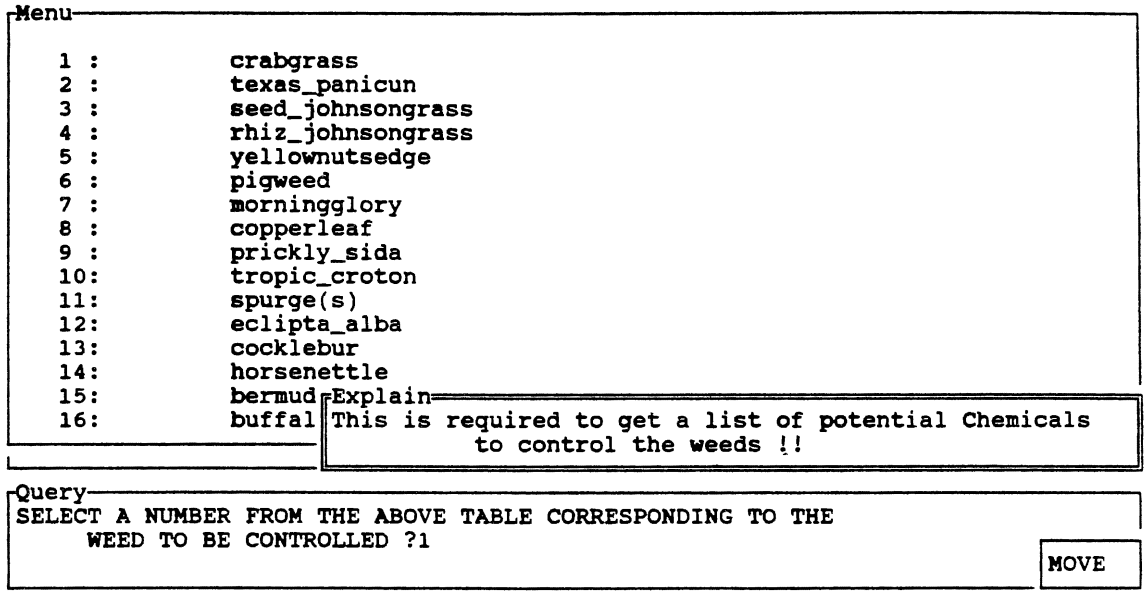

Figure a8. This figure explains how a window can be moved. Comparing figure a7 and a8, it can be seen that the explanation window is moved from the top right corner to bottom. This can be achieved by the default key F5 and the arrow keys.

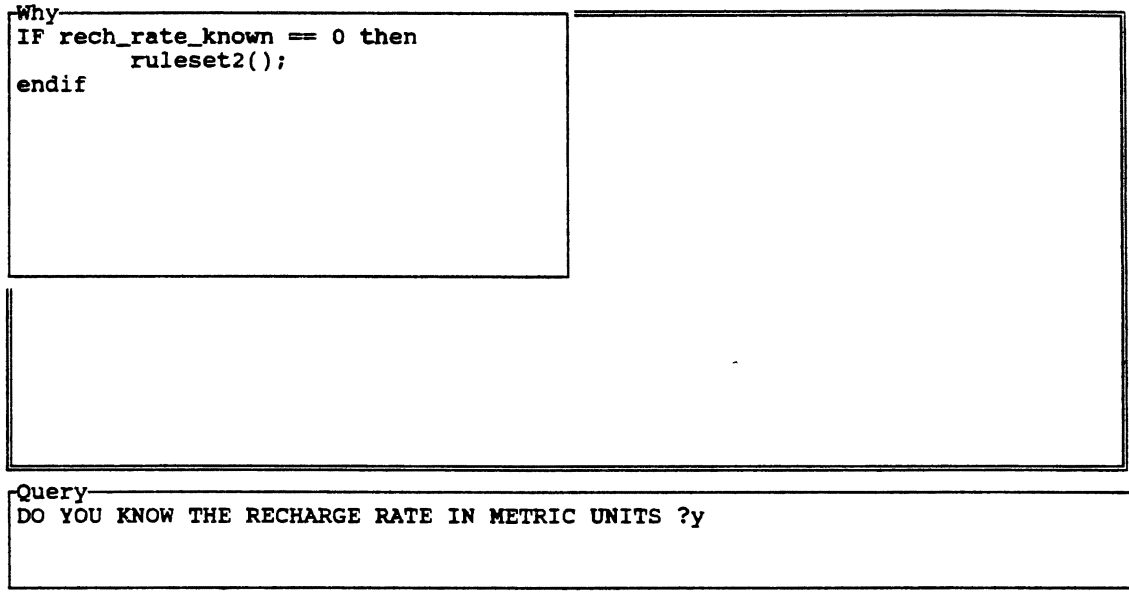

Figure a9. This figure explains how the recharge rate is fixed. If the user responds to the query as No, then the recharge rate is set based on certain rules, which is explained in Chapter III.

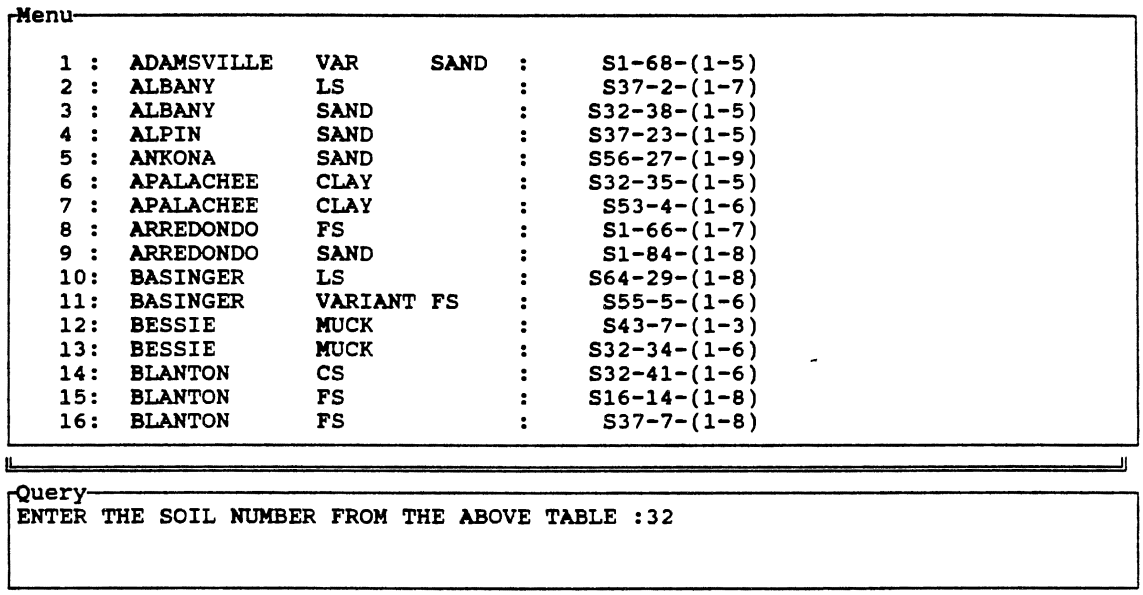

Figure a10. This figure explains how a user can select the type of soil. It is possible to scroll this screen by the use of arrow keys and the PgUp, PgDn keys.

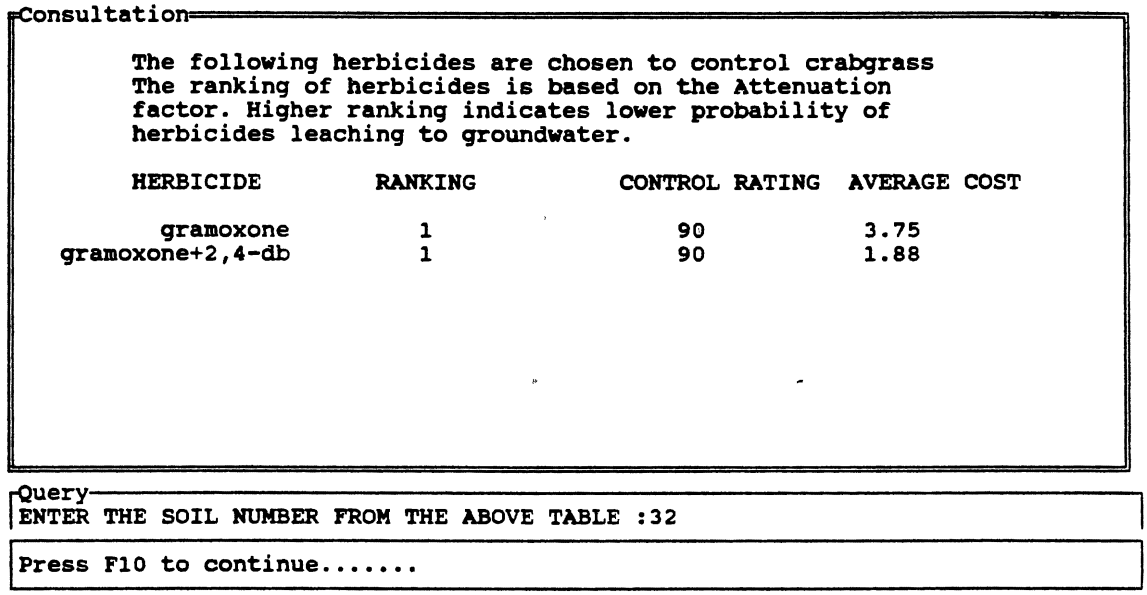

Figure a11. This figure explains a typical results obtained by a session.

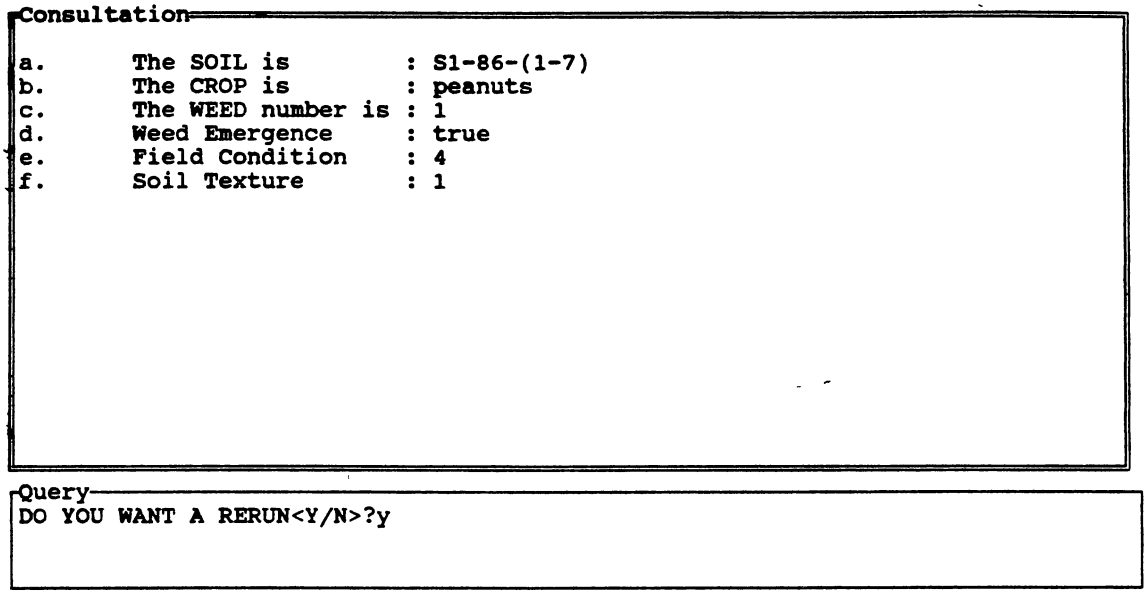

Figure a12. This figure explains how a rerun can be made. It allows the user to change only those variables. If any variable is changed, all the needed parameters associated with those variable is instantiated.

APPENDIX B

HARDWARE REQUIREMENTS

HARDWARE REQUIREMENTS

This section explains the requirements to run this expert system.

- It runs on IBM PC computers/compatibles, including XT, AT, and PS/2.
- Needs VGA card, any 80 column monitor.
- One floppy disk drive.
- To run the expert system the following files are required:

CXPROC.KBC	CXPROC.KB	KEY_PROC.H
KEY_PROC.TTY	CXPERT.EXE	TTYL.LIB
PESTI.EXE	PEANUTS.DAT	HERBI.DAT
COST.DAT	KANS.DAT	INP.DAT
WEED#.TXT	CHEM2.DAT	CHEM1.DAT
CHEM.DAT	SOIL.DAT	INP.INX

- Insert the disk containing these files in drive A.
- Enter PESTI < hit return >.
- To come out of the session during the middle of execution enter CnTl End and enter Yes.

APPENDIX C

CxPERT FEATURES

CxPERT FEATURES

Introduction

This section explains the CxPERT features invoked from the keyboard. These features are required in order to get several explanations like why a particular query is being displayed, what are the associated information, and so on. Some of the built in CxPERT functions as well as how to develop new functions is explained below.

- Explain function: Whenever CxPERT displays a query, the user can get an explanation by pressing the default key F1. A typical screen explaining the function of F1 key is given in the figure c1.

- Why function: Suppose the user wants to know about why a question is being asked, or a conclusion is being made, the explanations for these questions can be obtained from why function F3. A typical screen explaining the function of F3 key is given in figure c2.

- Error function: Suppose a query is being displayed, then if the user responds by entering an illegal value, the error function displays the legal values those can be used for that particular attribute. It is also possible that the developer defines the limits of the values that are legal. This is explained in section 3.3.1.

- Help function: Suppose the user wants to know the operations on a window for example, like increasing the size of the window, vertical scroll, horizontal scroll, and so on, this can be obtained by pressing default key F4.

- Change background color function: Background color can be changed by using the default key F6. All the above said functions can be used during the execution

of the program.

Several predefined keys are available and are explained in [6]. It is also possible that the developer can add new hot keys into the existing predefined hot keys [6]. For example, if a function draw() is to be linked, then draw() is to be declared in key_proc.h. This can be achieved by the following code:

```
extern int draw();
```

If this function draw() is to be invoked by F10, say, then we need to place the function name in the F10 location as given below:

```
*key_proc[]
     ----
     ----
     ----
     ----
     error,    /*F7*/
     do_stats,/*F8*/
     help,     /*F9*/
     draw,     /*F10*/
     ----
     ----
```

After making the changes, recompile the main program. The new functions can then be accessed from the keyboard.

VITA

Balaji S. Holur

Candidate for the degree of

Master of Science

Thesis: A HYBRID EXPERT SYSTEM FOR CONTROL OF WEEDS ON CROPS

Major Field: Computer Science

Biographical:

Personal Data: Born in Mysore, India, Decembetr 12, 1959, the son of Bhagavan S. Holur and B.K.Indiramma.

Education: Graduated from S.J.College of Engineering, Mysore, in 1983; received Bachelor of Engineering in Mechanical Engineering; received Master of Technology from Indian Institute of Technology at Kanpur, India in June, 1986; Completed requirements for the Master of Science degree at Oklahoma State University in May, 1991.

Professional Experience:

- Teaching Assistant, Department of Computing and Information Sciences, August, 1988 to May, 1989.
- Student professional, Department of Agronomy, August 1988 to May, 1990.
- Member, R&D, TVS-SUZUKI Ltd., August, 1986 to June, 1988.