Development of POMME, the Pest and Orchard Management Expert System

by

Rajesh S. Virkar

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Department of Computer Science and Applications

APPROVED:

Dr. John W. Roach

Dr. Michael J. Weaver

Dr. Roger W. Ehrich

Dr. Charles R. Drake

December, 1984
Blacksburg, Virginia

DEVELOPMENT OF POMME, THE PEST AND ORCHARD MANAGEMENT EXPERT SYSTEM

by

Rajesh S. Virkar

(ABSTRACT)

Direct dissemination of expert knowledge to agricultural producers through computer programs will increase product quality as well as profit margin. The construction of an expert system to help farmers manage apple orchards is reported. The system provides advice regarding specific pest management, treatment of winter injuries, drought control and general pesticide selection. The knowledge structures employed in the construction of the system are explained, and some sample interactions are provided. A model of the apple scab disease cycle is incorporated into POMME to give the system a more fundamental reasoning capability than available from the use of infection tables. Extension experts who have run trial cases on the system have approved its release for use by commercial apple growers.

ACKNOWLEDGMENTS

First and foremost I would like to thank my advisor Dr. John Roach for his untiring patience and the valuable assistance he provided without which this work would not have been possible. I am especially grateful to Dr. Michael Weaver and Dr. Charles Drake for many extremely useful discussions throughout the period of this work. I would also like to thank Dr. Roger Ehrich who assisted me as a member of my advisory committee. Thanks are also due to Mrs. Lois Wade who always provided a word of encouragement.

I dedicate this thesis to my parents, Shrikrishna and Kumudini, my uncle Raghunath and my aunt Pramila.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

Chapter I

INTRODUCTION

In recent years, researchers in artificial intelligence have developed programs that reflect the decision making strategies of experts in different fields. The pattern of thought processes put forth by human experts and interpreted by computer scientists is closely followed while creating these programs and corresponds to two parts of expert level 'thinking': an expert level knowledge base, and information regarding the use of this knowledge. These programs are called "Expert Systems".

Expert systems have evolved into a major field for research and application. These systems have earned the name by demonstrating a capability for performing expert quality work in areas previously thought to be the exclusive domain of highly trained human beings. Expert systems contain the knowledge of a human expert in a narrow domain of endeavor such as structural elucidation of organic molecules (DENDRAL) [16], bacteriological infections (MYCIN) [30], lung disease (PUFF, CENTAUR) [2] and soybean disease diagnosis (PLANT) [17].

Recently, expert systems have been implemented using a technology new to computer science: rule-based programming.

Programs in this new technology consist entirely of rules where an individual rule has two parts, a premise and a conclusion. The premise typically consists of patterns that must match before the rule "fires." The conclusion may be either an action to be performed or an implication that the system uses in further reasoning.

Expert systems are "prescriptive", that is, they suggest solutions to problems by reasoning in a manner similar to human experts. Simulation modeling systems, with which expert systems are often confused, are "descriptive"; that is, they can describe the relationships between different components of a target system without the fundamental knowledge of that system required to recommend any solution. Both kinds of system incorporate models; in expert systems they are of an expert human being while in simulation systems they are of a target system under study. Expert systems are knowledge-based, and modeling systems are data-driven.

To focus our attention on a major problem tackled in this study, consider the following example and its analysis. Jack boards a train and leaves station A at noon. According to the time table of the train, it is assumed that Jack will reach station B six hours from the time the train left station A. It is past 6:15 now, and Jack has not arrived at station B. Our problem is to determine Jack's whereabouts and if possible, predict his arrival at station B.

There are many inferences that can be made from the reported situation. Some obvious ones can be listed as follows:

(a) The train was delayed owing to rain or a snowstorm.

(b) The train has arrived without Jack since he took the wrong train.

(c) Jack took the right train but alighted at a wrong station.

(d) Jack alighted at an intermediate station to run some errands.

(e) The train met with an accident.

If we have more knowledge about Jack and his journey, we may be able to carry out temporal planning using the cause-effect analysis and determine Jack's whereabouts. Knowing that Jack has been to station B before, we can probably rule out inferences (b) and (c). Similarly, if we have the weather report of the area through which the train passes, we can make a decision on the possibility (a) in determining Jack's location and the time of Jack's arrival at B. It is also possible to rule out the possibility of (e) if we have the traffic reports at our disposal. Knowledge of the nature of Jack's errands can also help us project the time of his arrival.

Thus, the importance of causal reasoning can be seen in predicting the outcome of a situation using multiple knowledge sources. The idea of causal models that makes use of the fundamental factors affecting a situation is incorporated in this study to express expert level thought processes.

The particular application that motivated the work reported here is the problem of apple orchard management. In remote areas of the country where a human expert cannot always be reached in time to analyze the situation, predict outcomes, recommend solutions, and save the crop, we believe that an expert system can be very useful in aiding the orchardist to manage his orchards. The usefulness of an expert system is magnified on an international scale, where the number of available experts is very small.

Although orchardists have enough general knowledge about fruit-growing to become successful growers, sometimes they must turn to technical specialists such as plant pathologists or entomologists for advice. Often such a technical specialist (an expert in a specialty area) is approached by a grower who describes the crop, its condition (which may include the symptoms of a disease or traces of insect infestation), stage of growth, the weather conditions (which may include temperature, relative humidity, rain, etc.) and the type of help he may be interested in. The task of the expert

is to choose a technique (possibly chemical control) that solves the problem of the grower. A grower's problems typically include preventing diseases, insects, weeds, winter injury (freeze), drought, frost and other factors from reducing the crop. The supply of food to the populace critically depends on the availability of such consultations to solve agricultural problems.

The uncertainty involved in the prediction of a disease is eliminated by always using the minimum threshold while predicting an infection. The reason for this being that the growers are interested in preventing an infection rather than diagnosing it.

The degree of difficulty of agricultural problems is raised by several factors. One such factor is the compatibility between any two chosen pesticides. Residual effect of pesticides is another factor that needs some consideration. If a certain pesticide is used year after year, then the strain of the pathogen (fungus, bacterium, etc.) or the insect tends to develop an immunity toward the pesticide. This also makes the problem of pesticide selection hard for agricultural experts.

POMME, (Pest and Orchard Management Expert System) has been developed as an aid to apple growers in situations where the expert must travel hundreds of miles to examine an

orchard. It selects the appropriate pesticide (or set of pesticides) or management technique when required information is supplied by the grower. The grower would normally consult POMME when unusual conditions exist in the orchard such as ten days of continuous rain.

The knowledge base of POMME has information about fungicides, insecticides, compatibility, freeze damage, frost damage and drought damage. A description of the structural primitives that make up the foundation of the knowledge base is provided in this thesis to facilitate a clear understanding. The knowledge for POMME's knowledge base was gathered from C. R. Drake (Department of Plant Pathology, Physiology and Weed Science, Virginia Tech) [8] and from Agrios [1] and Childers [5]. Relevant information was extracted through discussions with Dr. Drake and was then transformed into rules. Knowledge is represented in frames for the ease of access and execution. There are various different types of frames in POMME. A limited natural language capability is supplemented by a menu system to enhance the interaction between the user and the system.

The most significant advance reported in this thesis corresponds to the use of causal model of disease in determining the stage of the disease in its life cycle. The correct prediction and appropriate recommendation can be easily made

with the help of the fundamental knowledge of the life cycle of the fungus. An advantage of using a causal model is that it is region-independent; i.e., under similar conditions a causal model will give same results when applied in Virginia, Michigan or Washington.

GUESS is a general purpose expert system shell developed at Virginia Tech (Lee- M.S. in progress) and is continually undergoing enhancements. GUESS is a frame-based shell that provides different knowledge structures and a natural language capability. Currently, an effort is being made to transfer POMME to the framework of GUESS.

This thesis is organized into seven chapters. The next chapter gives the literature survey. Chapter three describes the use of Prolog and includes a discussion of the knowledge base of POMME and its declarative frame representation. Chapter four presents the idea of causal models. Chapter five describes all the subsystems of POMME and discusses its applicability. Chapter six presents GUESS and discusses its usefulness through the transfer of POMME to its shell. Chapter seven includes the conclusions that can be drawn from this study.

Chapter II

LITERATURE REVIEW

Artificial Intelligence is a relatively new branch of computer science. It lends itself to the development of intelligent computer systems that reflect intelligent properties of human behavior such as reasoning, learning, comprehension of natural language, etc. The emphasis of current research in artificial intelligence is on knowledge representations.

Different search methods have been devised in the last twenty-five years. Nilsson [21] provided an overview of blind search as it can be applied to the state-space problem representation and the "and/or" graph. The A*-algorithm used for optimal search for an optimal search was presented by Hart, Nilsson and Raphael [12,13]. For this algorithm, a start node and a goal node have to be specified in order to determine the minimal-cost path in the state-space problem representation. Pohl [23] provided a heuristic algorithm that carried out a bi-directional search. The forward search and the backward search were ordered, and used functions similar to those of A*. Nilsson [21], also provided an algorithm for heuristic search of an "and/or" graph. In this algorithm, the node expansion is carried out by first

identifying the most promising potential solution tree, and second, choosing a proper node in that tree.

Knowledge representation involves a combination of data structures and procedures that, when used properly, will exhibit intelligence. Several schemes have been developed to represent knowledge. Logic was applied to artificial intelligence by Green [11] in the QA3 system, and by Fikes, Hart and Nilsson [10] in the STRIPS system. The QA3 system utilized first-order logic, while STRIPS used first-order predicate calculus for knowledge representation.

Quillian [24], Norman and Rumelhart [22], and Anderson and Bower [3] put forth the idea of semantic networks for formalizing knowledge representation. A semantic network consists of nodes and arcs connecting these nodes. The nodes represent objects, concepts, or situations in the problem domain, while the arcs represent the relationships between the nodes. Meaning is given to a particular network by the procedures that are used to manipulate it.

Production systems that make extensive use of if-then type rules have been implemented in a number of different systems. Waterman [33] used production system to play the game of draw poker. This production system used heuristics to learn and play better from experience. Shortliffe [30] designed the MYCIN system as a production system. Other pro-

duction system implementations include Rychener [27] who reimplemented several systems using production system methodology.

Wilks [35] used semantic primitives in a natural language system for machine translation. He defined a primitive as a symbol that is used but not defined within the system. Using the foundation of primitive acts, Schank [28] formed the conceptual dependency theory. This theory proposes task independence; i.e., when input text is submitted, the following tasks are executed: (1) paraphrasing the text, (2) translation to another language, (3) drawing inferences, and (4) answering questions.

Minsky [18] presented the idea of frames for knowledge representation. Frames provide a facility to organize related knowledge. A frame has slots to store the expected information regarding a particular attribute. Frames were originally used for visual perception and natural language dialogues. Schank and Abelson [29] put forth the idea of scripts to represent knowledge. A given script provides a normal or default sequence of events, exceptions, and specification of error situations. Scripts also use static descriptions such as props and roles that may make use of frames.

As presented by Clark and McCabe [6], Clocksin and Mellish [7], and Roach and Fowler [25], Prolog is a logic programming language based on first-order predicate calculus. A program written in Prolog consists of axioms in first-order logic, and a theorem that is to be proved with the help of these axioms. The axioms are represented by implications in horn-clause form. The system uses automatic backtracking to return instantiations of the variables used in the theorem to prove the theorem true if possible. This backtracking can generally be controlled by the user. Prolog provides an inherent foundation for rule interpretation, and uses pattern matching to execute a chain of rules. With its data structures and control structures, Prolog is very useful in artificial intelligence applications.

The concept of developing expert systems in certain specialized areas is not completely new. Shortliffe [30] has described the development of MYCIN, an expert system for diagnosing bacterial infections in human beings and selecting proper therapy. MYCIN was one of the pioneering efforts in building expert systems. MYCIN carries on an interactive dialogue with a physician and can explain its reasoning if the physician so desires. It includes a knowledge acquisition and management subsystem called TEIRESIAS which allows the expansion or modification of the rule base. This rule

base is made up of hundreds of production rules. The attributes, objects, and values form a vocabulary of domain-specific conceptual primitives and these primitives are used to formulate inference rules. The system uses a backward chaining control structure that results in an exhaustive depth-first search of an "and/or" goal tree, and uses confidence factors to accomodate inexact reasoning. MYCIN was written in LISP, and drew its inferences from a "disease-if-symptoms" type knowledge base that was encoded in rules. This knowledge base was exhaustive but did not include any fundamental reasoning.

Aikins [2] has reported work on two expert systems, PUFF and CENTAUR, for diagnosing lung diseases. These two systems accomodated the concept of frames as put forth by Minsky [18]. The frames in CENTAUR were called prototypes and were supplemented with the property of hierarchical execution. Prototypes had slots for the values of attributes of different objects (as suggested by Minsky) and also had slots that worked as pointers to other prototypes. Such a hierarchy of frames has been implemented in POMME, the pest and orchard management expert system reported here.

Weiss, Kulikowski, and Safir [34] worked on an expert system called CASNET that was developed for medical diagnosis of glaucoma. The system represents a disease by a dynam-

ic process that can be viewed as a network of causally linked pathophysiological states. The diagnosis is performed by checking the pattern of causal pathways present in the patient, and matching it with a disease category.

Lindsay, Buchanan, Feigenbaum and Lederberg [16] were involved in the development of DENDRAL which has been used for the structural clarification of organic molecules. DENDRAL has a causal model implemented in it that determines the cause-effect relationships between different chemicals and uses it to predict chemical reactions. DENDRAL uses knowledge about mass-spectrometry, and data acquired from a mass spectrometer to define constraints on the number of molecules generated by the structure generator. The knowledge about mass-spectrometry is encoded in rules that are applied during planning (to interpret mass-spectral data and to infer molecular fragments) and in rules that are applied during testing (to simulate the action of the mass spectrometer on the proposed structures). This approach to the application of a causal model has been modified in the work reported here. We have applied a causal model not only to predict the appropriate reaction in a process, but also to determine the current stage of that process. Agrios [1] has discussed the life cycle of the apple scab causing fungal pathogen and has been used as the basis of the causal model in POMME.

Childers [5] provided other peripheral details of apple crop maintenance that have been implemented in POMME.

Michalski and Chilausky [17] have designed an expert system called PLANT for disease diagnosis in soybean. We have not built POMME to be a diagnostic system for apples since it is believed that diagnosis of a disease implies the existence of that disease, and existence of a disease can cause a lot of damage to the crop. We have changed the focus of the expert system to make it a predictive system that suggests preventive maintenance techniques.

Rudd, Ruesink et. al [26] have presented the systems approach to solving the insect control problems for soybeans. Soybean ecosystem models have been applied with the plant growth model and the population dynamics models of insects to determine the economic injury levels, and suggest optimal control for the insects. Beck, Johnson et. al [4] started work on FAIRS system that is used in Florida to provide agricultural advice on soybeans, citrus fruits and tomatoes. It is essentially a menu-driven information retrieval system that uses some knowledge regarding the identification of insects in the fields and provides pesticide selections accordingly.

Many tools have been developed to build expert systems. Nii and Aiello [20] devised an expert system shell called

AGE. AGE was used for generating and modifying hypotheses in CRYSALIS (a system used in the domain of protein crystallography by Feigenbaum, Engelmore, and Johnson [9]). EMYCIN is a shell described by van Melle [32] that utilizes the knowledge representation and control structure of MYCIN, and can be used to create an expert system in any domain. Trigoboff and Kulikowski [31] developed IRIS as a tool for building expert systems in medical domains. It uses semantic nets and production rules to represent knowledge. The nodes in the semantic net hold information about a patient, while the production rules control the information flow between the nodes.

Lee and Roach [15], at Virginia Tech, have reported the development of GUESS, a General pUrpose Expert System Shell that is modular, frame-oriented and easy to use. Though GUESS is built on a smaller scale, it incorporates the advantages of many earlier shells such as EMYCIN and AGE. An attempt has been made in this project to make use of all advantages of GUESS by recoding POMME in the GUESS framework.

Chapter III

THE KNOWLEDGE BASE AND THE FRAME REPRESENTATION OF KNOWLEDGE

3.1 INTRODUCTION

Although not suitable for number-crunching, Prolog is an ideal language for symbolic manipulation and lends itself very credibly to expert systems application. We have written POMME in Prolog to make use of its rule-directed abilities.

The knowledge base of POMME primarily consists of pesticides since they are the most popular means today (as opposed to biological control, etc.) to prevent crop damage from different pests. The knowledge base also includes some non-chemical control techniques. All these techniques are broken down into structural primitives for the reasons of ease of system control, maintenance and update. System update may be required over a period of time as dictated by the Environmental Protection Agency; EPA annually certifies chemicals for use in the field. There are many structural primitives defined in POMME, and they are used in conjunction with each other in Prolog rules.

The data structures used in POMME are frames. Frames are implemented to bind related information together. This allows the search in a large knowledge base to be well-focussed and makes the execution of the expert system fast.

3.2 USE OF PROLOG

Prolog, a logic programming language, is suitable for building expert systems because it embodies the principle that 'computation is inference, not just calculation'. It also permits a uniformity in representing facts about the subject matter of the expert system and the rules that govern these facts. Knowledge representation is made easier by the declarative nature of the Prolog rules. Using Prolog, the rules that trigger the execution of others can also be programmed with ease. Modifications of data and the deduction mechanism can be achieved easily due to Prolog's modularity. Virginia Tech Prolog/Lisp [25] was used to build POMME, the Pest and Orchard Management Expert System presented in this thesis.

Prolog is a language that is always undergoing development and hence, is gaining useful extensions. It is more applicable than LISP for an expert systems application, because it inherently provides the infra-structure of rules, while for LISP, a programmer has to build his own foundation for the interpretation of his rules.

The normal control structure of Prolog is used to conduct an exhaustive depth-first search of the "and/or" goal tree, which is the basic structure of POMME's knowledge space. The rules encoding agricultural knowledge are divided into four

different partitions relevant to the sub-goals pursued by the system. Each partition is responsible for solving a class of problems such as pests, freeze, frost, and drought, where no two classes overlap.

3.3 STRUCTURAL PRIMITIVES

POMME's knowledge base includes a group of pesticides (fungicides and insecticides), their properties such as rates and compatibilities, and other non-chemical care techniques for the treatments of freeze, frost, drought, etc. All of these are controlled by a set of "Structural Primitives."

A structural primitive can be defined as the smallest piece of knowledge. The structural primitives are used to specify the logical relationships in the rules. The properties of structural primitives are: (a) Independence, (b) Non-Circularity, and (c) Primitiveness. Independence property states that no structural primitive depends on any other structural primitive. Non-circularity suggests that no two structural primitives can be defined in terms of each other. Primitiveness means that a structural primitive cannot be broken down into smaller pieces.

The structural primitives used in POMME can be explained as follows:

(1) SITE: This refers to the crop. All of POMME's knowledge pertains to apples. Future extensions may cover other fruit crops such as peaches and grapes.

(2) TIME OF SPRAY: The sprays in this set correspond to the time of the year (and stage of growth on the tree) and usually have a gap of seven days between consecutive sprays, according to label requirements and expert knowledge.

(3) CONTROL: This indicates the type of pesticide. For example, fungicides to prevent diseases, insecticides to control insects, herbicides to control weeds (not included in POMME), etc.

(4) RATES OF PESTICIDES: This includes spraying rates for all pesticides in the knowledge base. There are many ways of expressing spray rate such as pounds per hundred gallons, ounces per hundred gallons, fluid ounces per hundred gallons, pints per hundred gallons, pounds per acre, gallons per acre, pints per acre and ounces per acre.

(5) COMPATIBILITY: Compatibility of all pesticides is expressed through negative knowledge by encoding all pairs of pesticides that are not compatible with each other. These pairs are automatically accessed by the system when more than one pesticides are chosen by the user.

(6) CALIBRATION OF SPRAYER: This indicates the current spraying properties of a given sprayer. This information is collected interactively from the user by the system.

(7) EXPECTED TEMPERATURE: This is the expected temperature in the 24 hours after the proposed time of spray. It is advised that pesticides not be sprayed when temperature is very low (i.e., less then 35F), or very high (i.e., more than 85F).

(8) RELATIVE HUMIDITY: This is the expected relative humidity in the 24 hours after the proposed time of spray. It is advised that pesticides not be sprayed when relative humidity is very high (i.e., more than 90%), or rain is expected.

(9) RELATIVE TIME:

(a) The number of hours of wet foliage since the first green apple tissue is exposed in the spring. This information is used in monitoring apple scab disease development and prediction.

(b) The number of days passed since the last spray application. This information is used to determine the residual effect of a previous pesticide spray.

(c) The length of wetting periods during the growing season. This information is used to predict the timeframe and severity for an infection for diseases such as apple scab and cedar apple rust.

(10) CONDITIONAL TEMPERATURE:

(a) The air temperature during the wet period since the first green apple tissue is exposed in the spring. This information lends itself to determining the infection period and is usually used for apple scab.

(b) The air temperature during the wet periods of the growing season. This information is used for monitoring the infection periods during late spring, summer and early fall.

(11) DAMAGE FROM FROST: These are the different types of damage that can be sustained from frost. They are used for selecting the appropriate method to reduce the crop damage.

(12) KINDS OF LAND: These are the different types of land (aspect) on which an orchard can be established. This knowledge is used to select the proper technique of irrigation.

(13) KINDS OF SOIL: These are the different types of soil in the orchard, such as sand, gravel, clay, etc. This knowledge is used for determining the appropriate irrigation method.

(14) GENERAL CLIMATIC REFERENCE: The general climate of the region where the orchard is situated. It can be specified as arid, humid or moderate, and is used for finding solution to the drought problem.

(15) CONDITIONS OF WINTER INJURY: There are several kinds of winter injury, and they are expressed by various conditions of bark, cambium and wood. This knowledge is used to find ways to take care of the trees.

(16) DISEASES: The diseases that cause reduction in apple production. The set holds following eleven diseases: apple scab, cedar apple rust, quince rust, powdery mildew, black rot, bitter rot, fly speck, sooty blotch, Brooks' spot, storage rot and apple scald.

(17) INSECTS: The insects that can infest apple orchards. The set holds following twelve species: San Jose scale, aphids, mites, European red mite eggs, leaf rollers, leaf hoppers, red banded leaf rollers, codling moth, curculio, bud moth, green fruitworm and leaf miner.

All these structural primitives are used together in different combinations (with different values) to form Prolog rules. An example rule is given in Figure 1.

This rule invokes the following rules: Block, Cedar-Apple-Rust, Quince-Rust and Powdery-Mildew. It means that if Block BL3 is satisfied AND at least one of the two Rust rules is satisfied (i.e., control is desired for one of the two rusts) AND the Powdery-Mildew rule is not satisfied (i.e., control is not desired for Powdery Mildew), then assert the name of fungicide Mancozeb into the global database for possible use. Blocks are explained in the next section.

```
((USE MANCOZEB-80W) IF
          (BLOCK BL3)
          ( OR (CEDAR-RUST) (QUINCE-RUST))
          ( NOT (POWDERY-MILDEW))
          (ASSERT ((GLOBAL USE MANCOZEB-80W)) ))
```

Figure 1: Example of a rule

3.4 FRAMES

Frames [18] are data structures containing associated information about an object or a concept. Example frames include concepts such as "birthday party" (cake, kids, fun and games, etc.) and "weather" (temperature, relative humidity, wind speed, wind direction, etc.). A frame may include information about how to use the frame; it may contain information about reasonable ranges of values of attributes and the consequences if attributes have unexpected values. Frames typically have slots for values to be determined dynamically from the user or from deduction. The use of frames in POMME is illustrated below.

The knowledge structure employed by POMME for focusing the Prolog tree search on the grower's problem is that of 'Blocks'. The concept of blocks can be viewed as a declarative interpretation for the frame concept. Aikins [2], reports research on an expert system for lung disease using hierarchical frames. In POMME, there are three types of blocks: (a) weather blocks, (b) infection blocks, and (c) facets. A block contains a group of conditions that the system tries to satisfy interactively with the help of the user. A block may lead hierarchically to another block.

The infection blocks determine the validity of apple scab infection and depend on the number of hours of wet foliage

during early spring and the temperature during this wet period. Once the validity of this infection is determined, the facets are used to drive the pesticide selection. Facets depend on the type of pest control requested and the number of days passed since the last spray application. When the first block is not satisfied, the Prolog system automatically searches for another block that fits the data that are provided. The search continues until a suitable block is found.

3.5 EXAMPLE AND EXPLANATION OF BLOCKS

An example showing the slots and grouping of related knowledge in hierarchical blocks is provided in Fig. 2. An example showing the implementation of the hierarchy of infection blocks and facets is in Fig. 3. (variables are denoted by asterisks and comments are denoted by semi-colons).

This example shows that when the frame FACET F3 is invoked, it checks first to see if eradicative control is requested. Then it invokes the frame INFECTION IN2 which goes into its knowledge base to match the corresponding number of hours of wet foliage and the air temperature during that wet period. If frame INFECTION IN2 is satisfied, the number of days passed since the last protective pesticide spray application is determined. If this number is less than 8.5, frame

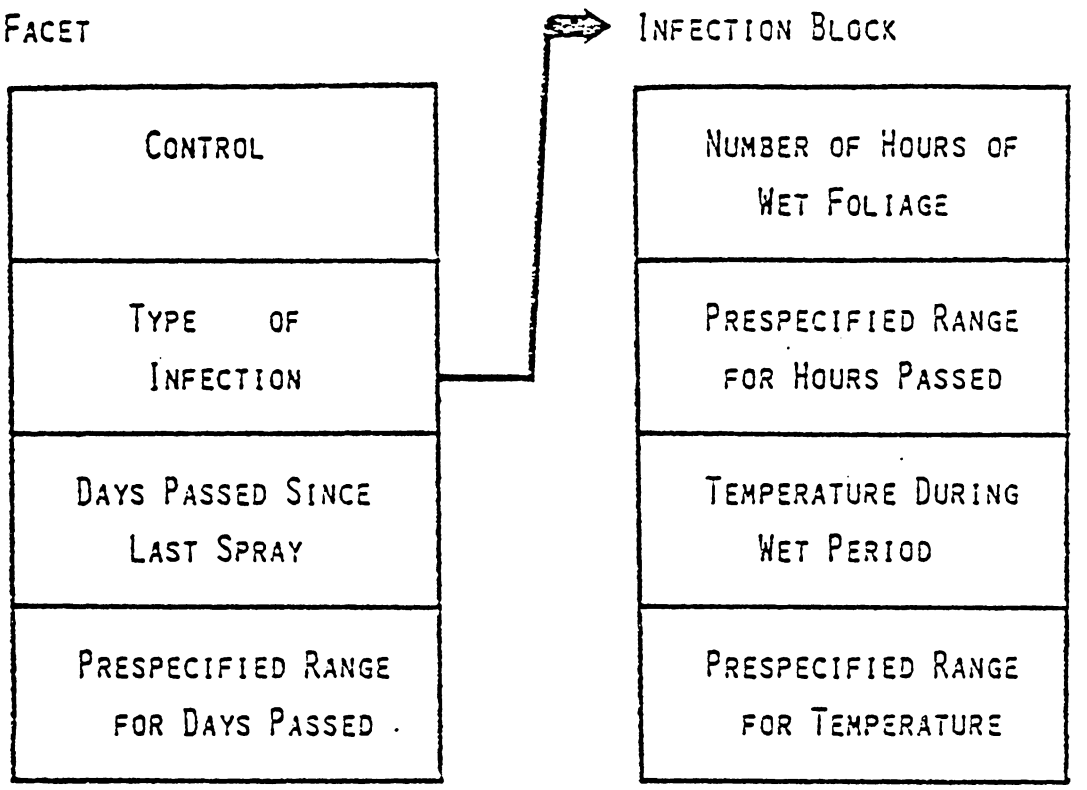


Figure 2: Scheme for implementation of the hierarchy of blocks

```
((FACET F3) IF
          (CONTROL ERADICANT)
          (INFECTION IN2)
          ; Days passed since last spray application
          (DAYS *C)
          (<= *C 8.5))

((INFECTION-BLOCK IN2) IF
          ; Number of hours of wet foliage
          (NUMHRS *A)
          (>= *A 30)
          ; Temperature during the wet period
          (TEMP *B)
          (>= *B 40) (< *B 42))
```

Figure 3: Hierarchy of blocks

FACET F3 is satisfied and is able to drive the selection for applicable pesticides.

POMME keeps track of the questions it asks a user so that the question need not be repeated when the system is exploring a part of the knowledge base needing a solution to the same problem. For example, when the system fails at INFECTION while satisfying a facet block, it does not ask the user questions about CONTROL while attempting to satisfy the next hypothesized block.

Blocks are also used in rules that choose the applicable pesticides. These rules are valid if the blocks are satisfied. The design of blocks is such that only one block will be satisfied in a given case. This focuses the search for applicable pesticides, helps prune the search tree and saves time.

3.6 KNOWLEDGE OF POMME

The knowledge of POMME is stored in several different fashions. Some of the knowledge is stored in assertions, some of it is stored in blocks, some of the knowledge is represented by functions, while some of it is stored in the rules that govern the other three knowledge sources. No attempt has been made to separate the knowledge from the controlling rules.

As mentioned before, the rules control the total structure of all the knowledge bases. Figure 4 shows the structure of these knowledge bases, and how they interact with each other.

There are different types of knowledge in the assertions. These types correspond to knowledge regarding the rates of pesticides, lack of compatibility between pesticides, efficacy (or efficiency) of insecticides against different insects, etc. Each of these knowledge bases has a different format, and a sample of assertions is provided in Figure 5.

Sections 3.4 and 3.5 provide detailed discussion about knowledge stored in blocks. Reference to Figure 3 is suggested for another glance at blocks.

Functions are used in different places in POMME. Situations where use of functions can be found are computing mixing ratios for pesticides to be mixed in spray tank, computing the number of days before spraying begins against San Jose scale, etc. Figure 6 gives sample functions used for mixing purposes.

Rules also are a source of knowledge. Knowledge in rules is seen through their power and flexibility to execute in a manner suitable to the gathered information. Many different examples can be provided here from each subsystem to show how rules represent knowledge. Figure 7 gives examples of rules that are chosen from the San Jose scale subsystem.

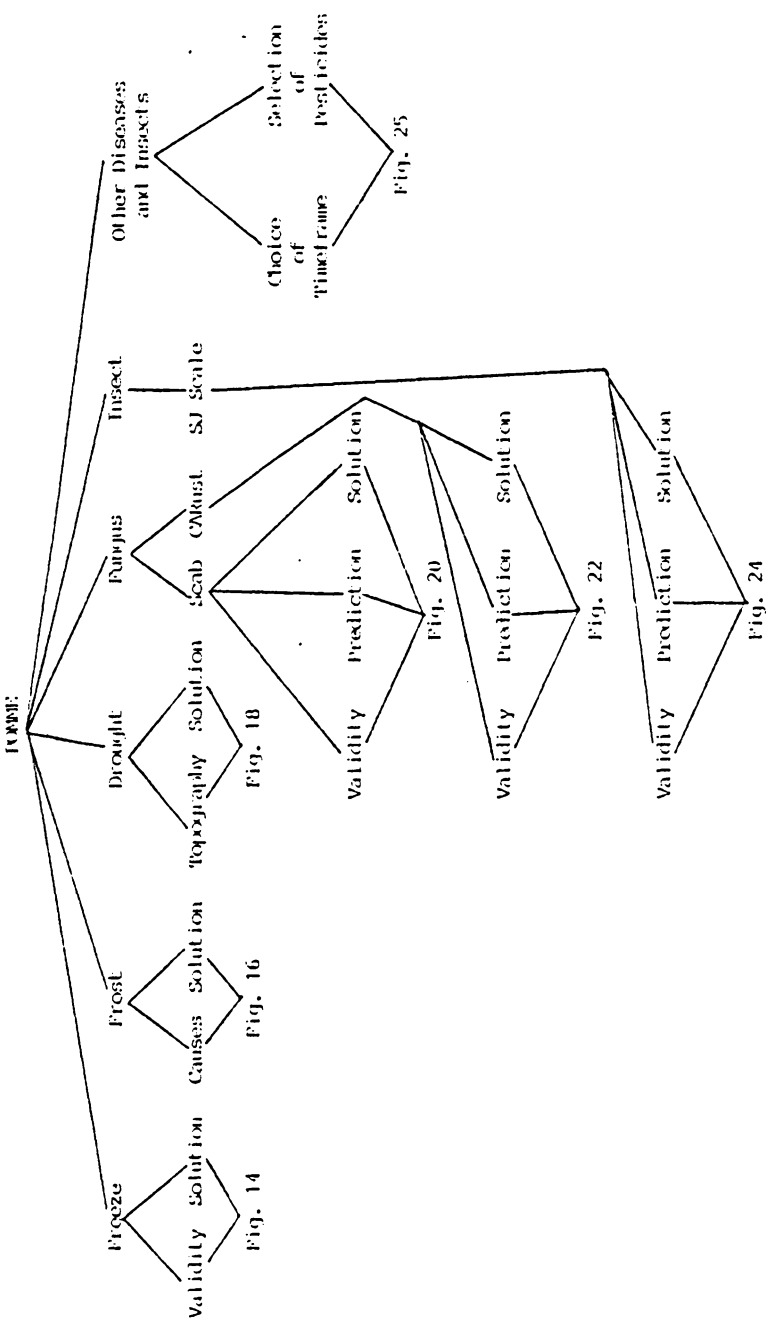


Figure 4: Structure of the Knowledge Bases

```
Rates of pesticides

(( invpest mancozeb_80w "Mancozeb 80W"
           (("Mancozeb 80W" lb 6.5) . nil) ))
(( invpest cap_thi "Captan and Thiram"
           (("Captan" lb 5.0)
            ("Thiram" lb 3.5) . nil) ))
---
---

Non-compatibility

(( dodine_65w methomyl ))  -----
(( polyram_80w funginex_182ec ))  -----
(( dikar_76_7w phosphamidon ))  -----
---
---

Efficacy of insecticides

(( control carzol a E ))  -----
(( control cygon a G ))  -----
(( control diazinon a G ))  -----
---
---
   [Note: "a" stands for Rosy Apple Ahids
          E  stands for Excellent
          G  stands for Good            ]
```

Figure 5: Knowledge stored in Assertions

```
   ; compute volume of pesticide

( ( begin4 *a *ac *c (( *p *q *r).*t) ) if

   ; unit for the pesticide is 'pints'

       ( == *q pt )

   ; convert it to gallons

       ( = *m ( * *ac *r ))
       ( = *z ( / *m 8 ))
       ( = *n ( + *c *z ))
       ( invunit *q *u )
       ( write *p " ---> " *m " " *u )
       ( begin4 *a *ac *n *t) )

   ; no more pesticides to be mixed

( ( begin2 *ac *c ) if
      ( NOT ( adam use *a ))
      ( global tank_size *x )
      ( write "The volume of chemicals=" *c "gallons.")
      ( = *y ( - *x *c ))
      ( write "Water occupies " *y " gallons." ) )
```

Figure 6: Knowledge stored in Functions

```
   ; check the recently read temperature; if it is
   ; in the range 20-100, then assert it and proceed

( ( cktemp *y *temp *x ) if
            ( <= *temp 100 )
            ( > *temp 20 )
            ( assert ((global temp_s *y *temp)) )
            ( = *z ( + *y 1 ))
            ( ask_temps *z *x ))

   ; if it is out of range, make sure that there is
   ; no error made in entering the temperature, assert
   ; the temperature and proceed

( ( cktemp *y *temp *x ) if
            ( OR ( > *temp 100 )
                 ( <= *temp 20 ))
            ( app_temp )
            ( assert ((global temp_s *y *temp)) )
            ( = *z ( + *y 1 ))
            ( ask_temps *z *x ))

   ; if there is an error, prompt for the same
   ; temperature

( ( cktemp *y *temp *x ) if
            ( OR ( > *temp 100 )
                 ( <= *temp 20 ))
            ( NOT ( app_temp ))
            ( ask_temps *y *x ))

   ; question to be asked

( ( app_temp ) if
  ( findit app_temp "Is this temperature correct?"))
```

Figure 7: Knowledge in Rules

Chapter IV

CAUSAL MODELS FOR ENHANCING EXPERT REASONING

4.1 INTRODUCTION

The fundamental factors that react with each other during the execution of a process are much more important than the average environmental conditions that exist during the course of the process. We have developed a causal model of apple scab disease that reflects this interaction between fundamental factors such as the development of the fungal pathogen and reaction of the host.

A causal model is more precise than the data tables that are generally used to determine infection periods. The reason being that tables are created from averages of very few variables that represent the environmental conditions affecting the disease process. The causal model is region independent since these fundamental factors always interact in the same fashion.

The causal model for apple scab implemented in POMME shows a second, deeper level of reasoning. It is invoked only when the data tables expressed in rules and blocks fail to achieve a satisfactory conclusion.

4.2 CAUSAL MODEL DEFINITION

Human experts typically have fundamental knowledge of their domain, including causal knowledge. This causal knowledge lets the expert interpolate and extrapolate on the state of any process in his or her domain. In the domain of this expert system, however, plant physiology and the development of diseases constitute that fundamental knowledge.

We have supplemented the knowledge base of POMME with a causal model of apple scab disease. By "causal model" we mean knowledge that captures the fundamental factors determining the course of the disease in a plant (or an entire orchard under study). For POMME, this knowledge is derived from field studies of the interaction between a host and a fungal pathogen.

The causal model simulates the growth of the pathogen under given conditions and the physiological reaction of the host. Simulation is important because it allows the knowledge of exceptional conditions to be used in determining the progress of a disease.

Application of the causal model is significant because the model is useful in deducing the validity of infection for particular rather than average conditions, and in case of uncertainty, it can lead the user by asking more specific questions and come to a conclusion. The causal model of di-

sease is applied whenever rules do not cover for the ailment of a plant. That is, if the symptoms and the weather conditions do not lead to a diagnosis, then the disease model is consulted for a prediction.

In contrast to disease models of POMME, MYCIN, an expert system developed for diagnosis of bacterial infections in humans, has a shallow representation of the disease process. Rules of MYCIN cannot solve problems not covered by its disease-if-symptoms knowledge base, nor can it account for new diseases such as Legionnaire's disease.

4.3 PROBLEMS WITH DATA TABLES

Data tables are products of field studies. When field studies are reduced to tables, exceptional conditions are left out: tables only capture knowledge about "average" conditions. These average conditions, nonetheless, are established over a long period of time and are valid for a particular area (which bears the orchard under study).

The inherent problem that lies with the averages can be seen from the following example. Consider the set S = {1,3,5,7}. The average of the values in set S is 4 which, of course, does not belong to the set S. Also, it does not supply any information about the set S itself. In a case where the data points follow a normal distribution (an example in

agriculture would be the temperature on a particular day), a range containing the average value can be specified to give more information about the set. This solution, however, is not very accurate because of the non-constancy of variables representing the conditions in an orchard.

Let us expand the given example as it is applied to our specific domain of plant diseases. A typical data table is formed with columns of 'average' values of temperature and corresponding relative humidities with number of hours of wet foliage required for a particular infection (e.g. apple scab). Now, a situation frequently encountered by a grower is that of unmatched columns. The grower may not have the temperature that matches the hours of wet foliage, or the number of hours of wet foliage to match the relative humidity or the temperature; and he/she may still wind up with an infected orchard.

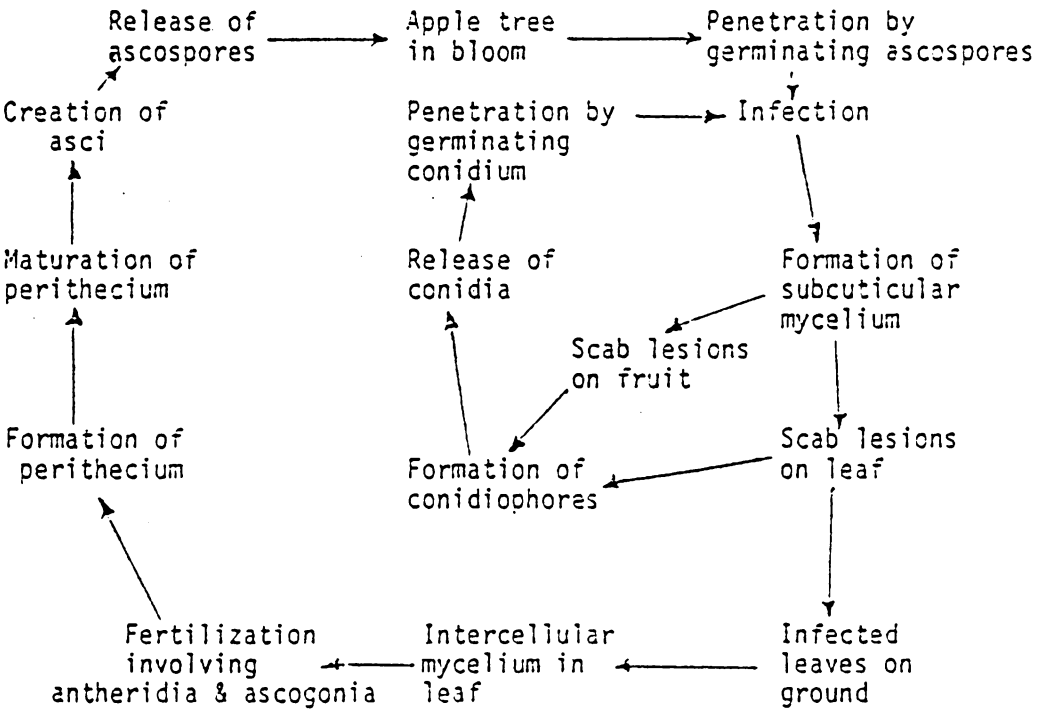
It is obvious that if there exist any conditions not covered by the tables or rules, prediction becomes impossible. We propose to use a causal model when the application of data tables fails. The data tables are created for superficial management of diseases and insects. MYCIN, with its disease-if-symptoms cross correlation rules, reflects the strength of data tables and thus, shows a superficial knowledge of bacterial infections.

The advantage of using a causal model is evident since the data tables contain observations that pertain to specific geographical regions and a causal model does not differ from one geographical region to another. For example, the apple scab fungus will always take the same path of development in its life cycle even though the data tables required for its monitoring will be different in Virginia than in Michigan or Washington State.

4.4 EXAMPLE AND EXPLANATION

Apple scab is the most important disease afflicting apples, and it exists in all apple producing parts of the world. It affects the apples by reducing the quality of fruit, the fruit size and the length of storage time for infected fruits. Other effects include premature fruit drop as a result of an infected stem, and defoliation as a result of severe leaf infections.

The symptoms of infection occur on leaves as well as fruits. At first, they appear as small, irregular, olive-colored spots on young leaves of the flower buds. Later, these spots are turned into lesions that are more circular in outline, metallic black in color and slightly raised. After infection, leaves may become curled and fall off. Because of early infection, the fruit may become misshapen and cracked.



(adapted after Agrios)

Figure 8: Graphic sequence of the life cycle of apple scab fungus

The lesions on fruit rupture the cuticle of the fruit at their margins.

These symptoms represent the growth of the fungal pathogen (Venturia inaequalis) in its life cycle. This life cycle is shown in Figure 8. There are different stages in the life cycle of this fungus and they are grouped into sexual and asexual stages. The sexual stages take the fungus from the decaying leaves on the ground to the production of ascospores that germinate on the leaves and on the fruits. The asexual stages include the formation of conidiophores on the leaves and the fruits, and the germination of conidia on the leaves and on the fruits.

The transcript of an interaction in Figure 9 shows the causal model of disease for apple scab in action. Most growers would need assistance for some of the below-listed questions.

When a comparison is made between the interaction above and the data table given in Figure 10, it can be seen that more variables play a part in the causal model than are included in the normal rule knowledge base or in infection tables [5]. While determining whether the primary apple scab infection has taken place, if conditions put forth (e.g., eight hours of wet foliage) do not lead to an exact diagnosis, then the causal model is invoked. The causal model

```
##### APPLE SCAB SUBSYSTEM #####
Have you had a primary apple scab infection ?
yes/no/unknown >>> unknown

##### DETERMINATION OF PRIMARY SCAB INFECTION #####
Please enter the number of consecutive hours of wet
foliage since the infection period began .....
 >>>  8

##### ENTER THE CAUSAL MODEL #####
Please enter the period of year :
 >>>  spring
Have the spores started germinating ?
 (Yes/No) >>> no
Has any leaf or fruit been infected yet ?
 (Yes/No) >>> no
Have the ascospores landed on any tree ?
(Yes/No) >>>  no
Have the ascospores been released into air yet ?
(Yes/No) >>>  no
Are the dead leaves on ground thoroughly soaked ?
(Yes/No) >>>  yes
Are the fruit buds open ?
(Yes/No) >>>  yes
Have the perithecia matured yet ?
(Yes/No) >>>  yes

  In this stage, the ascospores are
 released into the air.
  Thoroughly soaked perithecia lead to elongation
 of asci. The asci come out through the ostiole
 and discharge the ascospores into the air.
  Release of these matured ascospores into the
 air is necessary for the next stage.
-------------------------------------------------
  The probability of an apple scab infection has
 increased at the conditions you have specified,
 and hence, a protective spray application must
 be made.
-------------------------------------------------
```

Figure 9: A sample interaction with the causal model

represents the life cycle of the apple scab fungus, "Venturia inaequalis (Cke.) Wint." (Fig. 8), and asks questions regarding visual signs depending upon the period of the year. The responses given by the user lets POMME determine the stage of the fungus in its life cycle. POMME then displays this stage as well as the conditions that must occur for the growth and the progress of the fungus.

The generalized model of the disease states that there are three essential factors for disease development:

(a) a susceptible host,

(b) an environment ideal for the fungus, and

(c) presence of the pathogen.

For example, a host is susceptible to apple scab during the early period of spring when the fruit buds are immature; an environment ideal for the apple scab fungus is the one that lacks preventive fungicides (or has fungicides that are rendered ineffective because of rain over a period of time); and the presence of the apple scab fungus is guaranteed when the foliage is wet due to rain for over twelve hours at a temperature of about 50F (10C). This model has been enhanced by including transformations in the plant and the fungus at the cellular level. These transformations include stages such as overwintering of the pathogen in dead leaves on the ground, ascospore maturation, ascospore germination and conidial infection [1].

Average Temperature	DEGREE OF INFECTION Light	Moderate	Heavy	Days Incubation[a]
°F	hrs.[b]	hrs.	hrs.	days
78	13	17	26	
77	11	14	21	
76	9½	12	19	
63 to 75	9	12	18	9
62	9	12	19	10
61	9	13	20	10
60	9½	13	20	11
59	10	13	21	12
58	10	14	21	12
57	10	14	22	13
56	11	15	22	13
55	11	16	24	14
54	11½	16	24	14
53	12	17	25	15
52	12	18	26	15
51	13	18	27	16
50	14	19	29	16
49	14½	20	30	17
48	15	20	30	17
47	17	23	35	
46	19	25	38	
45	20	27	41	
44	22	30	45	
43	25	34	51	
42	30	40	60	
33 to 41[c]				

[a] Approx. no days required for conidial development after primary scab infection.
[b] The infection period is considered to start at the beginning of the rain.
[c] Data are incomplete at low temperatures.
[d] From W.D. Mills, Cornell University.

Figure 10: Data table used for the determination of apple scab infection

The stages in the life cycle of the fungus Venturia inaequalis can be roughly divided according to the season of their occurrence. This division is 'rough' because often there is an overlap of stages in a given season. The user is first asked about the period of the year; this helps in determining the stage of the fungus in its life cycle since there are only a few stages that may be found in a given period. The mechanism of backward-linking is used to pinpoint a specific stage. This means that the user is asked questions that satisfy the preconditions for a given stage, and if the preconditions are not satisfied then the life cycle is traced backward until the resolution is made regarding some stage. When the preconditions are satisfied, a description of the current stage is also prepared, and with this description, further conditions are determined for the advancement of the fungus in the life cycle (this advancement is viewed from the disease point). Examples of rules that implement the causal model are given below.

The top level rule that fires when the causal model is invoked is called 'derive'. It calls 'period', which determines the period of the year, and 'stage', which matches the preconditions and determines the current stage and further conditions (see Fig. 11).

```
( ( derive ) if ( print " " " " )
                ( print "###ENTER THE CAUSAL MODEL###")
                ( print " " " " )
                ( period *p )
                ( stage *st *pcf *prenext )
                ( *st )
                ( *pcf )
                ( *prenext ) )
```

Figure 11: The DERIVE rule

There are ten 'stage' rules that are used for matching the seasons and also for matching the preconditions. One of them is provided below; the equivalence statements in the rule are used to establish the conditions that are already satisfied (*st), the current stage (*pcf), and the conditions that have to be satisfied for the fungus growth (*prenext) (see Fig. 12).

```
( ( stage *st *pcf *prenext ) if
                ( per st10 )
                ( pre st10 )
                ( == *st ( sta10 ))
                ( == *pcf ( pcf10 ))
                ( == *prenext ( prenext10 )) )
```

Figure 12: The STAGE rule

Chapter V

ILLUSTRATION AND DISCUSSION OF POMME

5.1 INTRODUCTION

POMME is an expert system whose top level is a menu driven structure. It is divided into various subsystems that can be accessed through this menu. The division of the subsystems is functional and reflects the goals all subsystems are trying to achieve. Some of the subsystems start execution automatically, depending on the user's responses but without the user's knowledge, and hence, do not appear on the menu.

The other subsystems can roughly be grouped into two sections; one for weather-damage recovery subsystems and the other for pest management subsystems. The first section takes care of winter injury, drought and frost problems, while the second section advises on the pesticide selections for apple scab, cedar apple rust, San Jose scale, and other diseases and insects.

This chapter also provides a sample interaction that shows the working of POMME.

5.2 ORGANIZATION OF POMME

POMME is divided into several subsystems that the user can access through a menu. These subsystems are for the treatment of apple scab, cedar apple rust, other biotic and abiotic (winter injury, drought) diseases, San Jose scale, and combinations of insects.

Apple scab, cedar apple rust and San Jose scale have been given more consideration since they are major causes for concern in the eastern U. S. apple belt. All of the subsystems are shown in Figure 4.

A user's access rights are checked when he enters the system. As the user proceeds to the menu, he is provided with a selection of eight different subsystems and the choice of exiting POMME. The chosen subsystem focuses attention on the given problem. The focusing mechanism used here is that of (1) hypothesize a solution subgraph from the AND/OR knowledge space that corresponds to the chosen subsystem, (2) conduct an exhaustive depth-first search of this subgraph, and (3) if the hypothesized solution is acceptable, then present it, otherwise go back to (1). The subsystem asks specific questions to determine the scope of the problem and the range of the solutions. After solving that particular problem the user is taken back to the menu system.

There are two other subsystems that are not mentioned in the menu, but work at a lower level and present themselves only when appropriate. One such subsystem is the "compatibility check" that executes on the non-compatibility knowledge base. Whenever a user is trying to solve a pest problem, POMME determines the selection of applicable pesticides and displays these pesticides one-by-one and lets the user make a three-way choice: (a) take a look at the next pesticide selection, (b) reserve a particular selection for intended use if the user is satisfied with it, and (c) quit. If the user chooses (b), then this subsystem is invoked to check the compatibility of this pesticide selection with all previously reserved selections.

The second subsystem is used for determining the mixing ratios. After reserving a pesticide selection and checking its compatibility, the user is asked if he/she is interested in finding out the mixing ratios for all the reserved pesticide selections. POMME gathers the information regarding the size of the spray tank and the rate of spraying from the user, queries its own knowledge base regarding the rates of the reserved pesticides, and returns with the mixing ratios of those pesticides, for the given spray tank with known spraying rate.

5.3 WEATHER-DAMAGE RECOVERY SUBSYSTEMS

There are three different subsystems that are used for recovery from weather damage. These correspond to the control of damage from frost, drought and freeze (winter injury).

When the "freeze" subsystem is chosen by the user (to treat freeze-related ailments), POMME asks questions regarding the state of the tree in the orchard and the weather to determine the validity and the extent of the injury, and then suggests ways for mending.

The state of the tree is defined by the condition of cambium, wood (sapwood and heartwood) and bark, and hence, questions regarding these variables are asked of the user. This subsystem has a top level rule called 'treat_freeze' that invokes 'caretake', which finds out the injury to the tree, and 'explore', which determines and presents a technique to salvage the tree accordingly. 'Initialize' clears the memory before going back to the menu. 'Treat_freeze' is given in Figure 13.

The structure of this subsystem is shown in Figure 14.

The "frost" subsystem, when selected, determines the causes of the damage and then proposes techniques to prevent and/or reduce the damage accordingly.

```
( ( treat_freeze ) if ( global choice 1 )
                      ( screen )
                      ( print "###FREEZE SUBSYSTEM###")
                      ( caretake )
                      ( explore )
                      ( initialize )
                      ( menu ) )
```

Figure 13: The TREAT_FREEZE rule

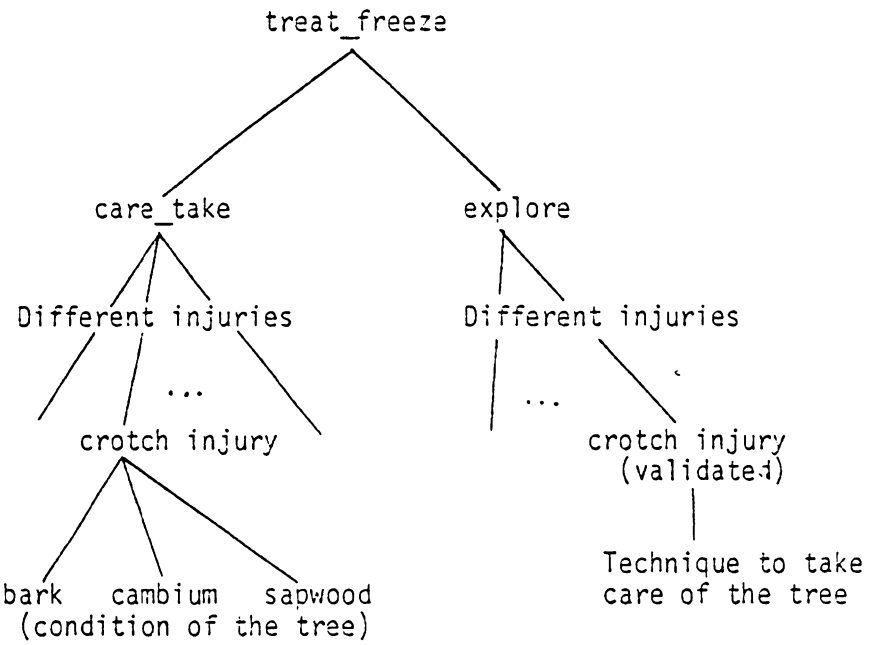


Figure 14: Structure of Freeze subsystem

Suggestions provided by this subsystem depend on various factors. These factors are radiation frost, layers of air with varying degree of temperature, robustness of trees, applicability of wind machines, and feasibility of a large number of heaters. The top level rule is 'treat_frost' (see Fig.15), and it invokes 'utilize', which in turn determines the solution, and 'present', which presents it.

The structure of the "frost" subsystem is given in Figure 16.

The "drought" subsystem, on the other hand, gathers information about the geography/topography of the orchard and then selects a method for preventing drought damage best suited for the particular orchard. Although trickle irrigation is the most popular method of watering today, it is not the best method for all situations. Here, knowledge regarding the proper use of the techniques is indicated.

'Treat_drought' is the top level rule here (Fig. 17) that calls 'make_it', which hypothesizes one solution at a time, and accepts or rejects it according to the topography it supports (and is presented at a given time). Factors such as general climate of the area and capability of soil to hold water are also taken into consideration.

The structure of this subsystem is presented in Figure 18.

```
( ( treat_frost ) if ( global choice 2 )
                     ( screen )
                     ( print "###FROST SUBSYSTEM###")
                     ( utilize )
                     ( present )
                     ( initialize )
                     ( menu ) )
```

Figure 15: The TREAT_FROST rule

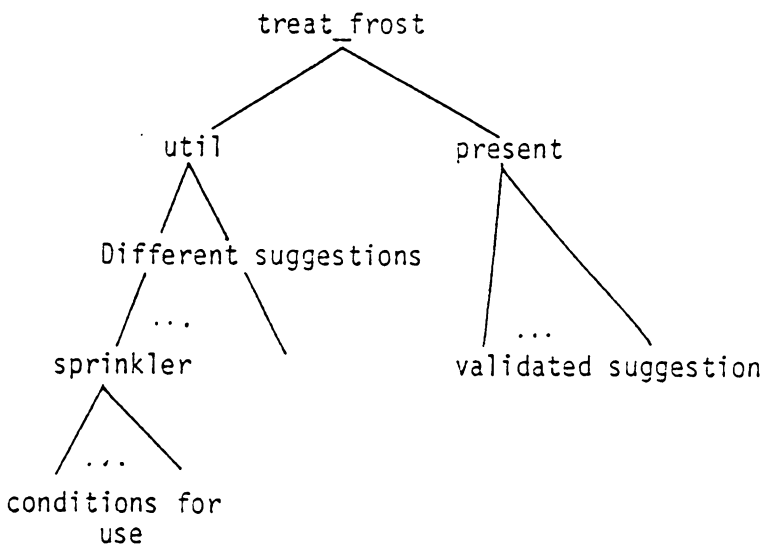


Figure 16: Structure of Frost subsystem

```
( ( treat_drought ) if ( global choice 3 )
                       ( screen )
                       ( print "###DROUGHT SUBSYSTEM###")
                       ( make_it )
                       ( prepare )
                       ( initialize )
                       ( menu ) )
```

Figure 17: The TREAT_DROUGHT rule

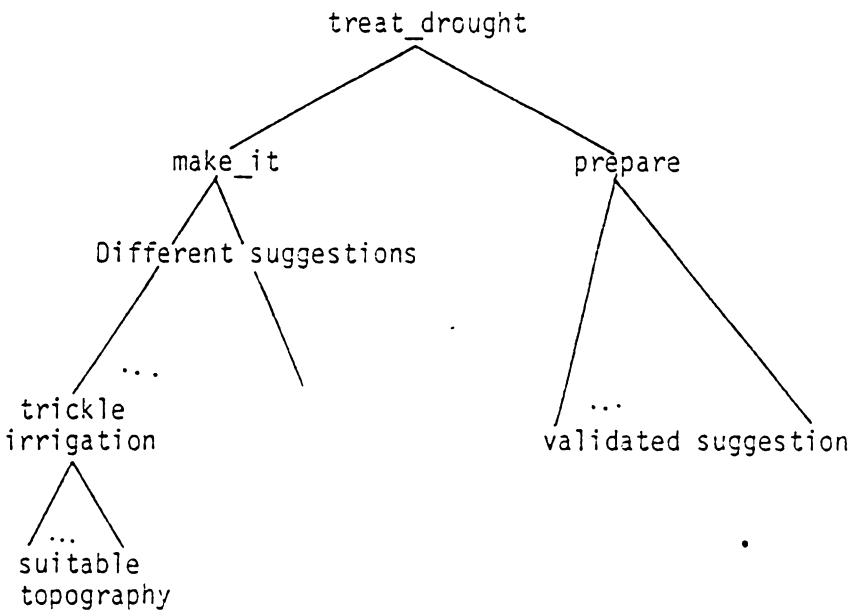


Figure 18: Structure of Drought subsystem

5.4 PEST MANAGEMENT SUBSYSTEMS

The subsystem devoted to apple scab determines whether information regarding the validity of primary infection is known, and if it is, whether primary infection has occurred. If the information about primary infection is not known, POMME determines the validity of primary infection, and from its conclusion goes on to predict secondary infection. The prediction capability of the system has been discussed in the chapter on causal models. If, on the other hand, the primary infection is known to have occurred, the user is taken directly to the prediction section.

The top level rule is called 'detas'. It invokes 'ask asprim' to gain some knowledge of primary infection, and then invokes 'detasstage' (determine apple scab stage) to determine infection and suggest appropriate fungicides. The top level rule is shown in Figure 19.

The structure of the apple scab subsystem is shown in Figure 20.

The cedar-apple rust subsystem is structured to determine the validity and predict the severity of the infection. A severe infection is defined as one that may cause economic loss against which a grower should have safety measures; a light infection is one that will not occur to a large extent and, hence, will not cause economic loss. Regular fungicide

```
( ( detas ) if ( global choice 4 )
               ( screen )
               ( print "###APPLE SCAB SUBSYSTEM###")
               ( ask asprim )
               ( detasstage )
               ( initialize )
               ( menu ) )
```

Figure 19: The DETAS rule

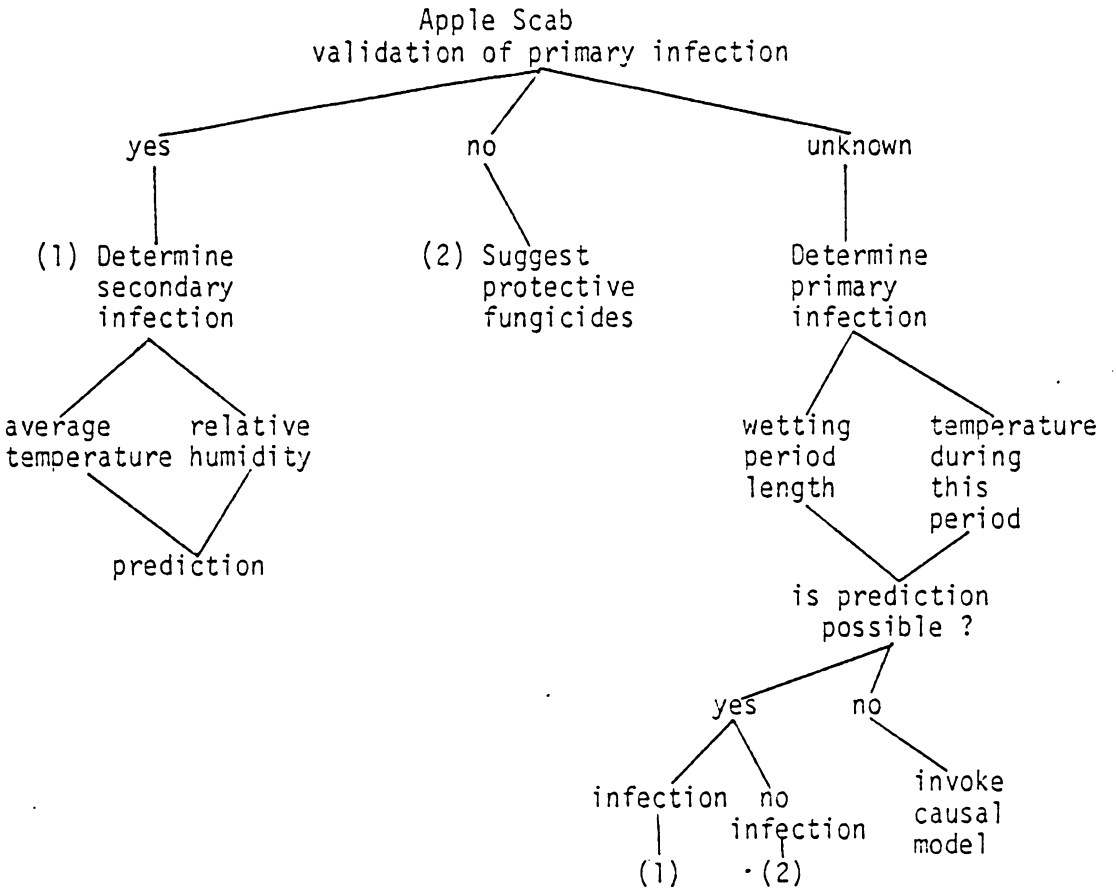


Figure 20: Structure of Apple Scab subsystem

sprays are viewed as a sufficient precautionary measure if a light infection is predicted; the prediction is followed by a selection of useful fungicides.

The highest rule that initiates this subsystem is called 'detcar' (determine cedar apple rust infection). It invokes other rules such as 'getmonth' and 'ginfo' to gather relevent information on current month and growth of the apples from the user. 'Chktimin' works like a function to determine the severity of the cedar apple rust infection. 'Spillcar' presents the advice on fungicides. 'Detcar' is shown in Figure 21.

The structure of the cedar apple rust subsystem is shown in Figure 22.

The subsystem for San Jose scale collects information about time and temperature, and passes it on to its prediction section, where it is used to predict a timeframe during which an insecticide should be sprayed in order to prevent the insect from causing economic damage to the crop.

The top level rule here is called 'treat_scale'. It invokes 'use_traps' to determine the use of pheromone traps, and 'catch_male' to determine the activity of the male of the species in the orchard area. 'Comp_sc_dd' works like a function in determining the timeframe for effective spraying against the San Jose scale, while 'take_scale' suggests the insecticides. 'Treat_scale' is shown in Figure 23.

```
( ( detcar ) if ( global choice 5 )
                ( screen )
                ( print "###CEDAR APPLE RUST###")
                ( getmonth )
                ( ginfo )
                ( chktimin )
                ( spillcar )
                ( initialize )
                ( menu ) )
```

Figure 21: The DETCAR rule

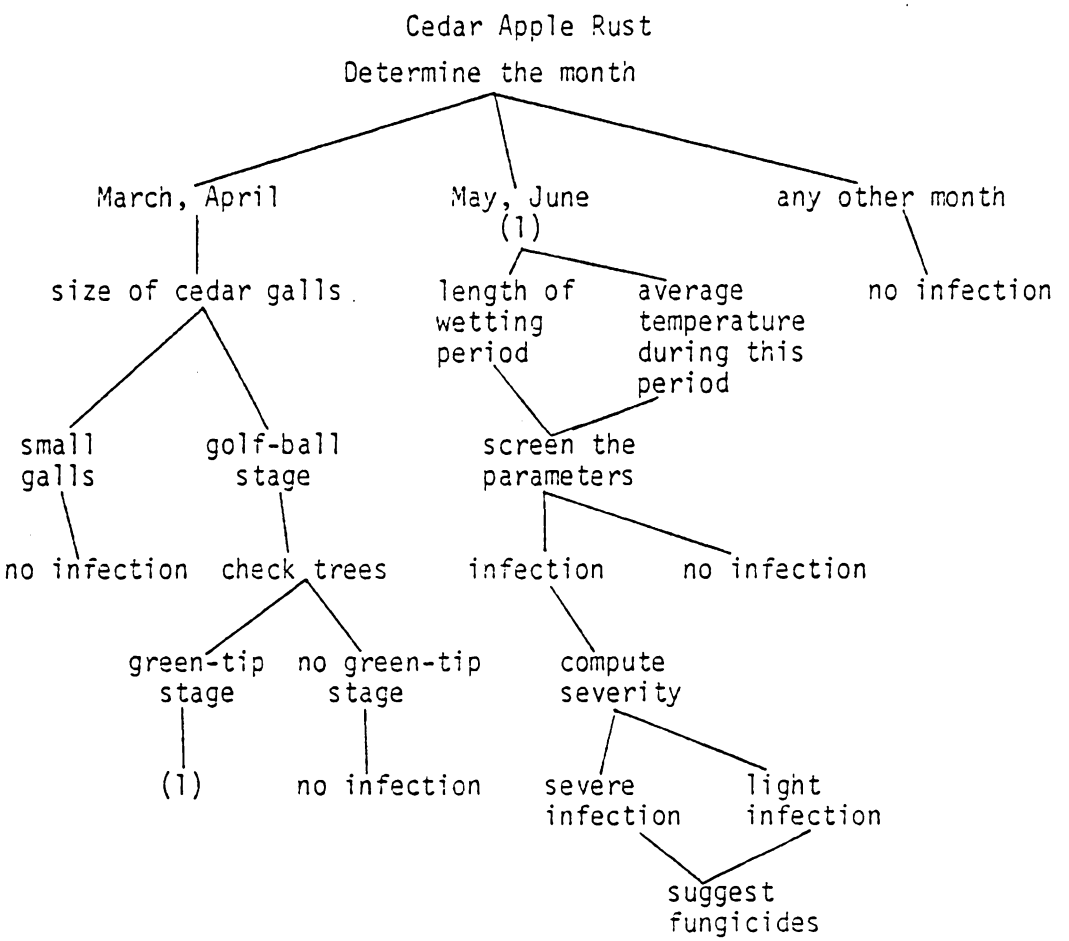


Figure 22: Structure of Cedar Apple Rust subsystem

```
( ( treat_scale ) if ( global choice 6 )
                     ( screen )
                     ( print "###SAN JOSE SCALE###")
                     ( use_traps )
                     ( catch_male )
                     ( comp_sc_dd )
                     ( scale_days )
                     ( take_scale )
                     ( initialize )
                     ( menu ) )
```

Figure 23: The TREAT_SCALE rule

The complete structure of this subsystem is provided in Figure 24.

Another subsystem is used to handle the day-to-day problems faced by growers, including precautionary measures for multiple diseases and/or insect combinations. This subsystem not only suggests chemical solutions, but also accepts or rejects timeframe of spraying depending on the known weather forecast.

The highest rule in this subsystem is called 'pesticides1' and asks questions regarding the time of the spray and the desired control (for example, fungicides, insecticides, or both). Then the subsystem is driven by the rule 'mixed', which uses information gathered by 'control'. 'Pesticides1' is given in Figure 26 for examination by the readers.

The general structure of the rules in this subsystem is given in Figure 25.

5.5 SAMPLE INTERACTIONS

Some sample interactions with POMME are given below.

```
### CEDAR APPLE RUST SUBSYSTEM ###

Please enter the current month
>>> march
Have the cedar rust galls reached golf ball stage?
(yes/no) >>> yes
Have apple trees reached green-tip?
```

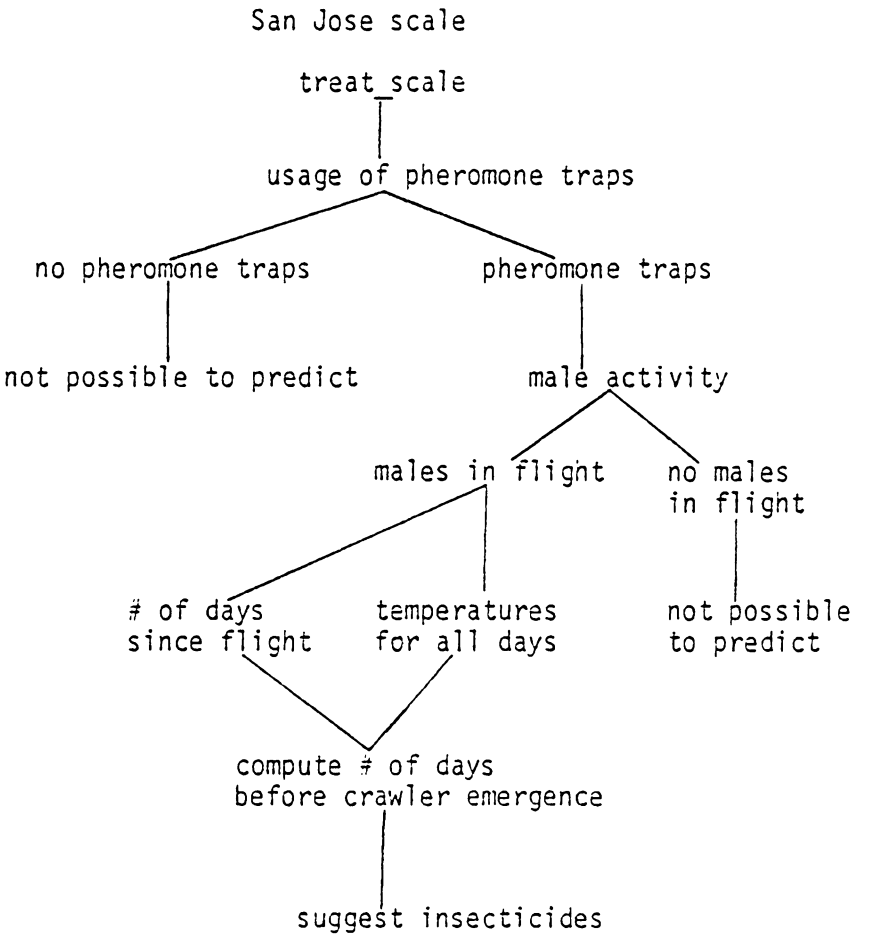


Figure 24: Structure of San Jose Scale subsystem

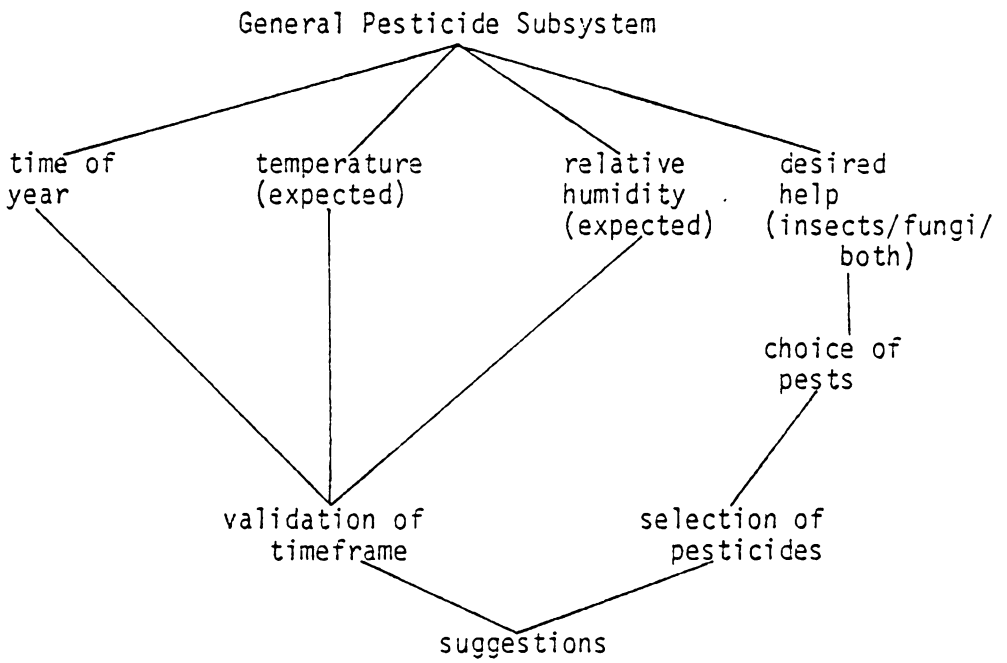


Figure 25: Structure of general pesticides subsystem

```
( ( pesticides1 ) if ( global choice 7 )
                     ( screen )
                     ( print "###GENERAL PESTICIDES###")
                     ( spray *spray )
                     ( control *control )
                     ( mixed ) )
```

Figure 26: The PESTICIDES1 rule

```
(yes/no) >>> yes
Please enter the length of wetting period
(in hours) >>> 15
Please enter the average temperature during
the wetting period
>>> 68

SEVERE Cedar Apple Rust infection can be
predicted. Use an eradicative spray as quickly
as possible in a time-frame with sunshine and
low humidity.

Would you like to have any recommendations for
cedar apple rust?
(yes/no) >>> yes

Here are a few fungicides used against cedar apple
rust---
--------------------------------------------------
Bayleton 50W should be used
1.5-2 oz./100 gal. dilute.
--------------------------------------------------
Please enter <a> for more
             <b> if you choose the selection
             <c> to quit
             >>> a
--------------------------------------------------
Funginex 18.2% EC should be used
10 fl. oz./100 gal. dilute.
--------------------------------------------------
Please enter <a> for more
             <b> if you choose the selection
             <c> to quit
             >>> a
--------------------------------------------------
Zineb 75W should be used
2 lb./100 gal. dilute.
--------------------------------------------------
Please enter <a> for more
             <b> if you choose the selection
             <c> to quit
             >>> b


### SAN JOSE SCALE SUBSYSTEM ###

Do you use pheromone traps in your orchard?
(yes/no) >>> yes
Have you caught any males in flight so far?
(yes/no) >>> yes
```

```
How many days ago did you first catch San Jose
scale in flight?
>>> 7
The average temperatures (in degrees Fahrenheit)
for these days were ---
1 >>> 68
2 >>> 67
3 >>> 70
4 >>> 63
5 >>> 59
6 >>> 58
7 >>> 71

--------------------------------------------------
The crawler emergence can be estimated to begin
in 16 days. That will be the right time to spray
eradicative insecticides.
This estimation would be more accurate if
temperature data is supplied for more days.
--------------------------------------------------
Would you like to know about insecticides that can
be used against scale?
(yes/no) >>> yes

Here are a few insecticides ---
--------------------------------------------------
Chlorpyrifos should be used
1.0 pt./100 gal. dilute and
2.5 pt. per acre.
--------------------------------------------------
Please enter <a> for more
             <b> if you choose the selection
             <c> to quit
             >>> a
--------------------------------------------------
Parathion 8E should be used
4.0 oz./100 gal. dilute and
12.0 oz. per acre, while
Superior Oil should be used
2.0 gal./100 gal. dilute and
6.0 gal. per acre.
--------------------------------------------------
Please enter <a> for more
             <b> if you choose the selection
             <c> to quit
             >>> b

Are you interested in tank-mix ratio?
(yes/no) >>> yes
Spray tank size (in gallons)
```

```
>>> 100
Total spraying rate (in gallons per acre)
>>> 250

Parathion --->     0.3 pounds
Superior Oil --->  2.4 gallons
Zineb 75W --->     2 pounds
The total volume of the chemicals is
2.4 gallons. Water occupies 97.6 gallons.


### FREEZE SUBSYSTEM ###

Is the heartwood darkened to a shiny brown color?
(yes/no) >>> no
Is the bark in the crotch area in dead state?
(yes/no) >>> yes
Is the cambium in crotch area in dead state?
(yes/no) >>> yes
Is the sapwood in crotch area in dead state?
(yes/no) >>> yes
Is the tree in a healthy state otherwise?
(yes/no) >>> no
Is the bark at or near the ground surface in
dead state?
(yes/no) yes

--------------------------------------------------
The tree is suffering from CROWN INJURY. Use
a 16-16-100 bordeaux mixture spray when buds
are swelling in spring at the delayed dormant
stage.
--------------------------------------------------


### GENERAL PESTICIDE SUBSYSTEM ###

Determine the spray needed ---
      Enter            For
        a      first pre-bloom spray
        b      second pre-bloom spray
        c      third pre-bloom spray
        d      fourth pre-bloom spray
        e      bloom-period spray
        f      petal fall spray
        g      1st cover spray
        h      2nd cover spray
        i      3rd cover spray
        j      4th cover spray
        k      5th cover spray
```

```
        l       6th cover spray
        m       7th cover spray
        n       8th cover spray
        o       post harvest spray
Please enter the spray
>>> a

Determine the type of control needed
       Enter     For Controlling
         a         Diseases
         b         Insects
         c         Diseases and Insects
Please enter the type of control
>>> a

What is the expected extreme temperature in the
24 hours following the proposed time of spray?
Please enter the low temperature if the high
temperature is not expected to exceed 85 deg. F.
>>> 75
Is the relative humidity expected to stay below
95% in the next 24 hours?
(yes/no) >>> yes
Do you need protection against cedar apple rust?
(yes/no) >>> yes
Do you need protection against quince rust?
(yes/no) >>> no
Do you need protection against powdery mildew?
(yes/no) >>> no

This is the first pre-bloom spray.
The timing of the first spray is the
green-tip stage, i.e., when the first
green tissue shows on the opening buds.
Here is a list of pesticides you can
use under the conditions you have
provided ---
-------------------------------------------------
Mancozeb 80W should be used
2.0 lb./100 gal. dilute and
6.5 lb. per acre.
-------------------------------------------------
Please enter <a> for more
             <b> if you choose the selection
             <c> to quit
             >>> c
```

Chapter VI

TRANSFER TO GUESS - GENERAL PURPOSE EXPERT SYSTEM SHELL

6.1 INTRODUCTION

GUESS is an expert system building tool developed at Virginia Tech. Its knowledge base consists of knowledge tables, knowledge trees and knowledge frames and provides for a communication blackboard. The inference strategies in GUESS are driven by action frames, programmer-defined functions and built-in library routines.

To make use of all the capabilities of GUESS, POMME was reorganized to fit the framework of GUESS. Knowledge stored in various different formats (including rules) in POMME was channeled into knowledge frames, trees and tables, and was separated from the control (represented by rules). The control was established through action frames and functions.

As a result of this experiment, new additions were made to the library routines of GUESS. Support was also provided for new data structures such as causal nets and bi-directional graphs. Dynamic creation and deletion of trees and tables had to be added to the routines of GUESS. POMME received features such as natural language support and consistent knowledge representations.

6.2 STRUCTURE OF GUESS

GUESS is a frame-based expert system building tool. It is projected that using GUESS's library routines and input/output environment, an expert system can be built with less effort, in less amount of time and with less problems regarding the modification (changing the logic or the knowledge base) of the system without introducing data errors or logical inconsistency. GUESS is continually undergoing enhancements. We will describe some background and the structure of GUESS before discussing the transfer of POMME to GUESS.

GUESS has been developed at Virginia Tech by Dr. John Roach and Newton Lee (Reference). The design of GUESS accomplishes many purposes such as system modularity, data integrity, data independence, localization of controls, explicit control flows, higher speed of execution and a framework for expertise modeling, and they are described below.

(1) System modularity: Implementation of information hiding is achieved and the system is made very adaptable to any modifications.

(2) Data integrity: The data integrity for all data bases is achieved through the implementation of a built-in security system.

(3) Data independence: Data independence is achieved through the separation of control from data. This permits

the modification of logic without affecting the data, and vice versa.

(4) Localization of controls: By supplying the property of readability, the localization of controls can be achieved since the control can be traced easily.

(5) Explicit control flows: The behavior of the system is made more obvious, predicatable, and hence, easily controllable.

(6) Reasonable speeds of execution: The turn-around time and the response time of the system are made relatively short.

(7) Framework for modeling expertise: This is achieved by providing a user-friendly, self-sufficient input/output environment for executing expert systems and frame-based representations of knowledge.

The knowledge base of GUESS is divided into four parts for information storage and communication. These parts are (a) knowledge tables, (b) knowledge trees, (c) knowledge frames, and (d) communication blackboard.

Knowledge tables are used for storing tabulated information. A knowledge table contains a set of tuples grouped together under a defined relation. Values in a tuple are accessed by keys defined uniquely in that table. The GUESS library routine TABLE_lookup is provided to access data stored in a table.

Knowledge trees are used for storing knowledge that shows natural hierarchy. A knowledge tree is a multi-way tree where each node contains a piece of a data element. A data element can be an atom or a list of atoms. TREE_lookup and TREE_expand are the GUESS library routines provided for manipulating data stored in trees.

Knowledge frames represent the knowledge of using and organizing knowledge stored in knowledge tables and knowledge trees as well as related knowledge gathered interactively from a user. It is called a frame because it is a data structure that contains associated information about an object or a concept. A knowledge frame typically has slots for values (as suggested by Minsky [18]) determined dynamically from the user or from deduction. There are five basic methods to fill in these slots: (1) looking up a table, (2) looking up a tree, (3) asking the user, (4) requesting another knowledge frame, and (5) using the default value. The fourth method listed above reflects the hierarchical structure of the knowledge frames.

The communication blackboard is a global area provided for the data communication among different processes. These processes include two types of frames, and assume sequential mode of execution. The blackboard has facilities for dynamic update and lookup, and is divided into three segments. The

knowledge frame segment stores information obtained by knowledge frames, the action frame segment stores information obtained by action frames (discussed ahead), and the object list segment stores the objects of interest referenced to by other processes.

A classification tuple that includes a security level and a category is applied to each knowledge base to implement the adapted security policy. The security levels are Top secret, Secret, Confidential and Unclassified. The categories include combinations of Read and Write accesses.

Action frames and programmer-defined functions are used in GUESS with the help of library routines to establish problem solving strategies. An action frame represents a goal to be accomplished. Actions performed inside an action frame are perceived as sub-goals and can be a combination of the following: (1) invoke another action frame, (2) invoke a knowledge frame, (3) invoke a programmer-defined function, and (4) invoke a GUESS library routine. Sub-goal (1) above reflects the hierarchy of action frames. A programmer is allowed to write his/her own routines to supplement the given library routines. These programmer-defined functions can be invoked from action frames or other functions.

The input/output environment is concerned with the processing of input strings; this includes tasks such as con-

version of all characters to upper case, removal of punctuation marks, performing spelling corrections, and so on. It is also concerned with providing interactive input/output facility.

6.3 TRANSFERRING POMME TO GUESS

After thoroughly understanding the frame-oriented capabilities of GUESS, POMME was reformed to suit the framework of GUESS. A primary task was that of separating the control from data because the original version of POMME was built on its knowledge base and no effort was made to keep the control flow independent of the information.

Hundreds of rules make up the backbone of POMME's control. These rules were reorganized and rewritten to meet the requirements of GUESS. Some rules were changed into action frames, while some were absorbed into knowledge frames and programmer-defined functions.

Action frames of level one represent different goals pursued by the system because they are called by the syntax-semantic action maps corresponding to the queries of the user. The new version of POMME has action frames for goals such as freeze treatment, drought treatment, determination of rates of pesticides, determination of compatibilities, finding pests controlled by a certain pesticide, and so on. A sample

action frame from the new version of POMME is given in Figure 27.

Knowledge tables in POMME represent things such as the rates of different pesticides and the units of different pesticides. This tabulated information was originally stored in a data base that employed a data structure of table with lists. A sample knowledge table of POMME is given in Figure 28.

A knowledge tree is employed in POMME to store information regarding the pesticide use. In the implemented two-level hierarchy, all the nodes in the upper level are pesticides while the nodes in the lower level are controlled pests. A part of this tree is shown in Figure 29.

Knowledge frames are generally used to gather information interactively from the user. A knowledge frame accomodates several question-asking rules (from the older version) that relate to the same goal. An example of a knowledge frame is given in Figure 30.

While recreating POMME, some functions were defined to manipulate the rates of pesticides. These functions were copied in semantic from the previous version, though the GUESS syntax was used. A part of a function is given in Figure 31.

Although the given facilities were able to handle most of the tasks, some problems needed more support. These problems

```
( (AF "the mixing ratio")
    ->
      (USAGE "find out the mixing quantities" LEVEL 1)
      (GETALL OBJECT_of_interest *a)
      (listlen *a *x)
      (>= *x 1)
      (KF "spray equipment parameters")
      (BLACKBOARD_lookup
           KF is "spray equipment parameters"
           OBJECT is nil
           KEY is tank-size
           VALUE is *sp_t)
      (BLACKBOARD_lookup
           KF is "spray equipment paramters"
           OBJECT is nil
           KEY is spray-rate
           VALUE is *sp_r)
      (:= *c 0)
      (print " ")
      (FUNC "start computing ratios" *a *c *sp_t *sp_r)
      (cut) )
```

Figure 27: An action frame

```
( (TABLE "rates of pesticides" UNCLASSIFIED (READ)
          (
               (MANCOZEB-80W          6.5)
               (DODINE-65W            1.5)
               (POLYRAM-80W           6.5)
               (THIOPHANATE-METHYL-70W      1.25)
               (CHLORPYRIFOS          2.5)
                ---
                ---

               (ZOLONE-3EC            16.0)
          )
) )
```

Figure 28: A knowledge table

```
( (TREE "pesticides and pests" UNCLASSIFIED (READ)
        (
           (MANCOZEB-80W     CEDAR-APPLE-RUST internal
                             QUINCE-RUST internal)
           (CEDAR-APPLE-RUST nil)
           (QUINCE-RUST       nil)
           (DODINE-65W       APPLE-SCAB internal)
           (APPLE-SCAB        nil)
            ---
            ---

           (DICHLONE         APPLE-SCAB internal)
        )
) )
```

Figure 29: A knowledge tree

```
( (KF "topography")
   ->
     (USAGE "find out the topography of orchard"
      LEVEL ANY)
     (or   (ASK_user
              question is "what is the structure of land"
              object is nil
              expect (leveled sloped rolling unknown)
              useless unknown
              answer is *l_a)
           (DEFAULT_of *l_a is unknown)
     )
     (cut)
      ---
      ---

     (cut)
     (BLACKBOARD_update
          KF is "topography"
          OBJECT is nil
          CONTENT is ( (land *l_a)
                       (type *s_a)
                       (climate *c_a)
                       (distribution *d_a)
                       (hold *h_a)
                     )
     )
)
```

Figure 30: A knowledge frame

```
( (FUNC "step 2 for ratios" *a *b *c *q *r *t *s)
  ->
   (USAGE "step 2 for ratios" LEVEL 2)
   (or (and (== *q pt)
            (:= *ac (/ *t *s))
            (:= *m (* *ac *r))
            (:= *z (/ *m 8))
            (:= *n (+ *c *z))
            (write *a " -----> " *m " pints.")
            (FUNC "start computing ratios" *b *n *t *s)
       )
        ---
        ---

   )
   (cut) )
```

Figure 31: A function

include manipulation of lists and implementation of the causal model. These problems were solved by supporting causal nets and graphs, and providing new list manipulation routines.

6.4 REMARKS

Transporting POMME to an expert system shell such as GUESS was a good exercise because it increased the applicability of POMME. As control and data were separated from each other, it became more obvious that POMME could be moved to any shell for testing the shell or POMME.

The most important task of knowledge engineering had been completed since POMME was already developed before this project began. Thus, the problem of testing the applicability (especially, the ease of use) of GUESS on a completely new venture still remains to be solved.

This transfer gave POMME some very important features. These features are a new format for knowledge representation, independence of control from data to aid in updating POMME, and a desired natural language capability. For GUESS also, this project brought some advancements as new features increasing its applicability were added to it.

Chapter VII

CONCLUSIONS

Agricultural products are grown throughout the world where growers need expert advice. Extension specialists typically provide individual consultations when possible. On an international scale, however, expert advice is limited because of the small number of experts available. Construction of computerized tools could help alleviate the problem of disseminating expert advice. The work reported here has been aimed at providing expert level guidance for maintaining apple orchards in remote areas. Apples are a valuable crop; the cash value of apples in the state of Virginia for 1984 alone was forty-eight million dollars.

Experts in plant pathology and entomology, who have closely watched the development of the system and have run trial cases on it, have testified to the accuracy of the system and agree that the system, when made accessible to the growers, will considerably reduce their workload.

Although trial cases have been run on POMME, true validation of POMME by growers and extension agents in the state has not been carried out because of some technical difficulties. The major roadblock is that of allowing growers and extension agents to access POMME. POMME currently resides on

a VAX 11/780 that is marked for research by the department of Computer Science, while extension agents and growers are generally permitted to access data bases on the IBM 370 on campus. Distribution of POMME on diskettes is ruled out because of POMME's size (the current executable code alone occupies over 250 KB of memory), and the problem of updating the system.

POMME is an expert system because it captures the knowledge of experts: plant pathologists, entomologists and others. POMME's ability to diagnose, predict infections and suggest solutions or pesticides shows expert care of orchards. Care has been taken to construct a system that orchard growers will find friendly and easy to use.

Prolog has been used to build POMME. This effort proves that Prolog is suitable for expert system application. The current system has well over five hundred rules and has reached a stage where it can be made accessible to the apple growers.

Breaking down the knowledge base into various structural primitives has proved to be a very helpful strategy in maintaining and updating POMME from time to time. Application of frames makes it easy to manage the grouped knowledge and focus the tree search. Dynamic creation and destruction of frames is an important area that could be looked into as further research.

Causal model application has been successful in determining the stage of growth in the life cycle of apple scab fungus. POMME makes an advance in expert system construction because it incorporates a causal model of disease not present in medical systems such as MYCIN. The model used here concerns itself with a well-defined process that has been studied for a number of years. Extension of the causal model concept will have an impact on determining which model an expert will choose under given conditions. Future research may involve causality as a function in modeling expertise.

Transferring POMME to the framework of GUESS has been an important project for POMME as well as for GUESS. It increased POMME's ability to modify and to incorporate new knowledge easily. POMME was also supplemented with natural language capability through this project. Many additions were implemented on GUESS as a result of this transfer. These improvements increased the applicability of GUESS.

Future work on POMME may include adding specific apple varieties, their uses, and eventually, extension to other crops. Dissemination of expert level knowledge to farmers will increase the quality of agriculture in the United States. POMME is one of the first of a new kind of computer program that will aid growers in the future. Expert systems

will realize the promise of revolutionizing the care of crops and farms.

BIBLIOGRAPHY

1. Agrios, G. 1980. Plant Pathology. New York: Academic Press.

2. Aikins, J. 1980. Prototypes and Production Rules: A Knowledge Representation for Computer Consultations. Report No. STAN-CS-80-814, Computer Science Department, Stanford University, Stanford, CA.

3. Anderson, J., and Bower, G. 1973. Human Associative Memory. Washington, D.C.: Winston.

4. Beck, H., Marlow, Jr., G., Pohronezny, K., and Johnson, F. 1983. Florida Agricultural Information Retrieval System; A Computerized Crop Management Program with Application to Tomatoes. Proceedings of Florida State Horticultural Society. 96:128-130.

5. Childers, N. F. 1976. Modern Fruit Science. Chicago: J. B. Lippincott Co.

6. Clark, K. L., and McCabe, F. G. 1984. Micro-Prolog: Programming in Logic. New Jersey: Prentice-Hall.

7. Clocksin, W. F., and Mellish, C. S. 1981. Programming in Prolog. Berlin: Springer-Verlag.

8. Drake, C. R. 1984. Personal Communication.

9. Feigenbaum, E. A., Engelmore, R. S., and Johnson C. K. 1977. A Correlation between Crystallographic Computing and Artificial Intelligence Research. Acta Crystallographica A33:13.

10. Fikes, R. E., Hart, P., and Nilsson, N. J. 1972. Learning and Executing Generalized Robot Plans. Artificial Intelligence 3:251-288.

11. Green, C. C. 1969. The Application of Theorem-Proving to Question-Answering Systems. IJCAI 1, 219-237.

12. Hart, P. E., Nilsson, N. J., and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on SSC. SSC-4:100-107.

13. Hart, P. E., Nilsson, N. J., and Raphael, B. 1972. Correction to 'A Formal Basis for the Heuristic Determination of Minimum Cost Paths.' SIGART Newsletter 37:28-29.

14. Kowalski, R. 1979. Logic for Problem Solving. New York: North-Holland.

15. Lee, N. S., and Roach, J. W. A General Methodology for Building Expert Systems: General pUrpose Expert System Shell. (unpublished).

16. Lindsay, R., Buchanan, B., Feigenbaum, E., and Lederberg, J. 1980. Applications of Artificial Intelligence for Organic Chemistry. New York: McGraw-Hill.

17. Michalski, R.S., and Chilausky, R. L. 1980. Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis. Policy Analysis and Information Systems. 4:(2).

18. Minsky, M. 1975. A Framework for Representing Knowledge. in P. H. Winston (ed.) The Psychology of Computer Vision. New York: McGraw-Hill.

19. Moto-Oka, T. (ed.). 1982. Proceedings of the International Conference on Fifth Generation Computer Systems. Amsterdam: North-Holland.

20. Nii, H. P., and Aiello, N. 1979. AGE (Attempt to Generalize): A Knowledge-Based Program for Building Knowledge-Based Programs. IJCAI 6, 645-655.

21. Nilsson, N. J. 1971. Problem-Solving Methods in Artificial Intelligence. New York: McGraw-Hill.

22. Norman, D. A., Rumelhart, D. E., and the LNR Research Group. 1975. Explorations in Cognition. San Francisco: Freeman.

23. Pohl, I. 1971. Bi-Directional Search. In B. Meltzer and D. Michie (eds.), Machine Intelligence 6. New York: American Elsevier, 127-140.

24. Quillian, M. R. 1968. Semantic Memory. In Minsky, 227-270.

25. Roach, J., and Fowler, G. 1984. The HC Manual: Virginia Tech Prolog. Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA.

26. Rudd, W. G., Ruesink, W. G., Newsom, L. D., Herzog, D. C., Jensen, R. L., and Marsolan, N. F. 1980. The Systems Approach to Research and Decision Making for Soybean Pest Control. In C. B. Huffaker (ed.) New Technology of Pest Control. John B. Wiley and Sons.

27. Rychener, M. D. 1976. Production Systems as a Programming Language for Artificial Intelligence Applications. Computer Science Department, Carnegie-Mellon University.

28. Schank, R. C. 1972. Conceptual Dependency: A Theory of Natural Language Understanding. Cognitive Psychology 3:552-631.

29. Schank, R. C., and Abelson, R. P. 1977. Scripts, Plans, Goals, and Understanding. Hillsdale, N. J.: Lawrence Erlbaum.

30. Shortliffe, E. 1976. Computer-Based Medical Consultations: MYCIN. New York: American Elsevier.

31. Trigoboff, M., and Kulikowski, C. 1977. IRIS: A System for the Propagation of Inferences in a Semantic Net. IJCAI 5, 274-280.

32. van Melle, W. 1980. A Domain Independent System that Aids in Constructing Consultation Programs. Report No. STAN-CS-80-820, Computer Science Department, Stanford University.

33. Waterman, D. A. 1970. Generalization Learning Techniques for Automating the Learning of Heuristics. Artificial Intelligence 1:121-170.

34. Weiss, S., Kulikowski, C., and Safir, A. 1977. A Model-Based Consultation System for the Long-Term Management of Glaucoma. IJCAI 5, 826-832.

35. Wilks, Y. A. 1973. An Artificial Intelligence Approach to Machine Translation. In Schank and Colby, 114-151.

The vita has been removed from the scanned document