83-24

83-24

Report No. UIUCDCS-R-83-1134

THE EXPERT SYSTEM PLANT/CD: A CASE STUDY IN APPLYING THE GENERAL PURPOSE INFERENCE SYSTEM ADVISE TO PREDICTING BLACK CUTWORM DAMAGE IN CORN

by

Albert Gerard Boulanger

July 1983

Report No. UIUCDCS-R-83-1134

THE EXPERT SYSTEM PLANT/CD: A CASE STUDY IN APPLYING THE GENERAL PURPOSE INFERENCE SYSTEM ADVISE TO PREDICTING BLACK CUTWORM DAMAGE IN CORN

BY

ALBERT GERARD BOULANGER

B.S., University of Florida, 1978

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1983

Urbana, Illinois

July 1983

DEDICATION

I dedicate this thesis to W.J., my wife and friend.

ACKNOWLEDGEMENTS

A system of this size could have not been developed by one person and I wish to thank all those who have participated in the design and building of ADVISE. First I wish to thank my thesis advisor, Professor Michalski. He originated many of the novel concepts in ADVISE which made working on it fun. He also suggested many useful changes to my thesis. I wish to thank Dr. Baskin for his help and suggestions. Both Professor Michalski and Dr. Baskin served very well as stewards of the ADVISE project. I wish to thank past participants of the ADVISE project now working in industry, Heinrich Juhn and John Davis. I wish especially to thank Bob Stepp for providing generous help and ideas even when he was no longer connected with the ADVISE project. I thank all those now currently working on the project, Lance Rodewald, Mark Seyler, Kent Spackman, and Carl Uhrik for their timely help that enabled me to finish before becoming an old man. I thank Dr. William van Melle at XEROX PARC for clearing up misunderstandings of EMYCIN. I thank John Reiter and Dr. Tim Nibblet for their support during bad times. I thank Professor Michie for introducing me to the field of expert systems. I thank the domain experts, Professor William Ruesink, Chip Guse, Steve Troester, and Ron Meyer for their patience while I tapped their minds. Steve Troester was tremendously helpful by explaining to me his Black Cutworm damage simulation programs, and letting me integrate them into PLANT/cd. Steve Briggs, Extension Specialist in Agricultural Entomology and the Natural History Survey and Ron Meyer, who was an Entomologist for the Natural History Survey during the work of this thesis, should be acknowledged for allowing me to use their unpublished data. Thanks go to Professor William Luckmann, Professor William Ruesink, Steve Troester, Ron Meyer, Professor R.S. Michalski, Dr. A.B. Baskin, Paul Baim, Chip Guse, Bob Reinke, Bob Stepp, and Carl Uhrik for reading, commenting, and suggesting changes to preliminary versions of this thesis. Paul Baim drafted Figure 5 and did not demand anything for doing it except some jaw breakers. He deserves one *attaboy*. I would also like to thank June Wingler for preparing Figure 14. The writing of this thesis started at the University of Illinois and was finished while working at Bolt Beranek and Newman Inc. in Cambridge Massachusetts. Thanks go to Dr. Al Stevens at Bolt Beranek and

Newman for providing resources to complete my thesis.

There were many tools that I used to produce this thesis. These included the PLATO system for producing some of the figures. The BRUNO package on the Department of Computer Science's Hewlett Packard computer was also used for drawing some of the figures. The thesis was typeset using TROFF and an Imagen laser printer. I thank the University of Illinois Computer Science Department for providing me with these facilities since if I where to do it with my hand, the thesis would look awful.

This research was supported in part by the Office of Naval Research grant No. N00014-82-K-0186, and in part by the U. S. Department of Agriculture grant No. 321512344.

TABLE OF CONTENTS

APPENDICES

CHAPTER 1

INTRODUCTION

This thesis describes the application of the general purpose inference system ADVISE to the creation of an expert system, PLANT/cd, that predicts damage to corn due to the Black Cutworm. ADVISE is a general purpose *meta*-expert system that allows the incorporation of a wide spectrum of domain areas and problem solving strategies. It has been developed by a group effort in expert system design and research. The ADVISE project is headed by Prof. Michalski and Dr. Baskin. Researchers currently working on various aspects of ADVISE are: Mark Seyler, Kent Spackman, Lance Rodewald, Carl Uhrik, Bob Reinke, and myself. Black Cutworm damage prediction is just one of the possible domains for ADVISE. ADVISE also has been used to build an expert system to diagnose soybean diseases, called PLANT/ds [Michalski et al. 1982], [Michalski et al. 1983c]. For a general overview of ADVISE, see [Michalski et al. 1983a] and [Baskin & Michalski 1981], and for a technical description see [Michalski et al. 1983b]. ADVISE was designed to be an adaptable expert system and this thesis is a case study in applying ADVISE to a particular domain.

Black Cutworm (BCW) damage occurs to a farming area on infrequent occasions because it is a sporadic pest. Although some fields consistently have cutworm problems, on the average only about 10% of the fields having damage in one year will be damaged in the next year. In any given year 2 to 10% of the Midwest corn acreage receives damage exceeding the economic injury level. A predictive expert system that identifies damage fields before planting would allow farmers to take action before planting and scout fields more effectively after planting. It is estimated that such an expert system could save one million dollars per year in the state of Illinois and five million dollars in the entire corn belt.

This thesis will first describe in general terms the ADVISE meta-expert system. It will then describe the cutworm damage problem and the implementation of PLANT/cd using ADVISE. It will be shown that this implementation brings up the issue concerning the connection of *surface* models and *deep*

models[1] (described in more detail in Section 3.5.) The following chapters provide detailed descriptions of the control scheme used for PLANT/cd, the implementation of the PLANT/cd knowledge base, some validating experiments, and a description of a small expert system, PLANT/cdp, implemented in EMYCIN that predicted fields that could be damaged. Finally, a summary section provides a discussion of the strengths and weaknesses of ADVISE as learned from its application to the specific problem of Black Cutworm damage prediction, and of the strengths and weaknesses of the PLANT/cd implementation. The next section describes briefly the concept of expert systems.

1.1. A Brief Review of Expert Systems

An expert system is a computer program that exhibits high performance in a specific problem domain due to a large amount of formally encoded knowledge and the ability to conduct formal reasoning on this knowledge. An expert system is designed to do various tasks that an expert would typically perform: diagnose, interpret, consult, classify, identify, search through a space of possible solutions, explain, tutor, and analyze.

In expert systems, domain knowledge is often represented as a set of production rules. These rules take the form of:

IF <condition> **THEN** <action>

where

<condition> is a two valued or variable valued logic expression which must be satisfied to a certain preset degree, and

<action> is a set of actions to be performed if the <condition> part of the rule is satisfied.

The set of such rules that characterize a given application area is termed the *knowledge base*. The knowledge base is one of the major components of an expert system.

Another major component is the inference mechanism by which the knowledge base is used to perform given tasks. An implementation of a method is called the inference procedure. Often the

[1] A deep model involves a precise, algorithmic formulation. A surface model involves a shallow, approximate, "rule of thumb" formulation.

inference procedure is divided into two parts:

- a *rule evaluator/executor* that applies and evaluates a *single* rule, and
- a *control scheme* that decides about the *order* in which rules are applied.

In many expert systems, some form of *probable* or *plausible* reasoning is used in the inference procedure to handle uncertainties. Data is often represented as *value/confidence pairs*.

In addition, an expert system includes the memory needed to store intermediate results of rules when they are actuated or *fired*. Some architectures term this component the *blackboard*.

A favorite starting point for expert system architecture is to organize these three components, i.e., knowledge base, rule evaluator, and blackboard, as a production system. Such a production system organization works by performing *recognize-act* cycles. In each cycle, the control scheme decides which rules to evaluate and, if any fire, executes their right-hand sides. For more details on production systems see [Nilsson 1980] , [Davis & King 1976] and [Michie 1980]. Not all expert systems work as pure production systems. Some, and PLANT/cd is one example, are more like Markov algorithms in that the productions are ordered.

There is another way to view the architecture of an expert system that turns out to be equivalent in many respects to the view of an expert system as a production system. In this view, the expert knowledge is in the form of a network. The control scheme becomes a network traverser/updater. This is the view incorporated in PROSPECTOR [Duda et al. 1978]. For more detailed overviews of expert systems, see [Gevarter 1982], [Michie 1980], and [Stefik et al. 1982].

CHAPTER 2

THE ADVISE SYSTEM: A GENERAL OVERVIEW

2.1. Novel Features of ADVISE

ADVISE has been designed to provide the knowledge engineer with an expert system workbench. It contains a number of features not commonly found in existing expert systems. These features include:

- facilities for representing knowledge in three different forms: a network form, a rule form, and a relational data base form,
- use of a very general and flexible representation for inference rules,
- freedom to choose and apply a host of inference control strategies,
- inductive learning capabilities,
- freedom to choose and apply several rule evaluation schemes,
- separation of control of flow specification *(strategic information)* from non-procedural information *(tactical information)*,
- modular design,
- common virtual memory representation for data,
- emphasis on simplifying man-machine interaction, and
- implementation of the system in Pascal (a popular language widely available on many computer systems).

Many of these features will be discussed in the next section. As this list shows, there is an emphasis in ADVISE on user and developer flexibility and convenience. The user interface has been designed to be *friendly* and is designed around a touch panel interfaced with a plasma display terminal. The user interface also supports a variety of other terminals as well. Figure 3 shows some views of the plasma

display terminal.

There is the design goal of getting ADVISE *out to the public.* This is a major reason why Pascal was chosen as the implementation language. Good versions of Pascal are available on many microcomputer systems. There are two methods to support microcomputers. One method is hand tailoring ADVISE to the microcomputer environment. The other more attractive option is automatic tailoring. A tool for ADVISE to do automatic tailoring is being developed.

There is a goal of a *clean design* in ADVISE. In many expert systems the tactical and strategic knowledge are intertwined. This causes problems in explaining the reasoning of the inference process to the user. ADVISE is being designed to avoid this problem by maintaining a separation between control or *meta*-knowledge, and *factual* knowledge of the domain. The goal of a clean design is also supported by a common representation for data at the lowest level of ADVISE. ADVISE was designed as a set of modules serving as elementary blocks for constructing inference systems. This modular design makes ADVISE flexible and elegant.

2.2. A First Look at the ADVISE Architecture

Figure 1 shows a conceptual level diagram of ADVISE. (For more detailed information see [Michalski et al. 1983a] and [Baskin & Michalski 1981].) The four major components of the ADVISE system are:

- Control Block and User Interface
- Knowledge Base
- Query Block
- Knowledge Acquisition Block

Each of these components supports a network structure, a rule base, and data base knowledge representations. These components are described in the following sections.

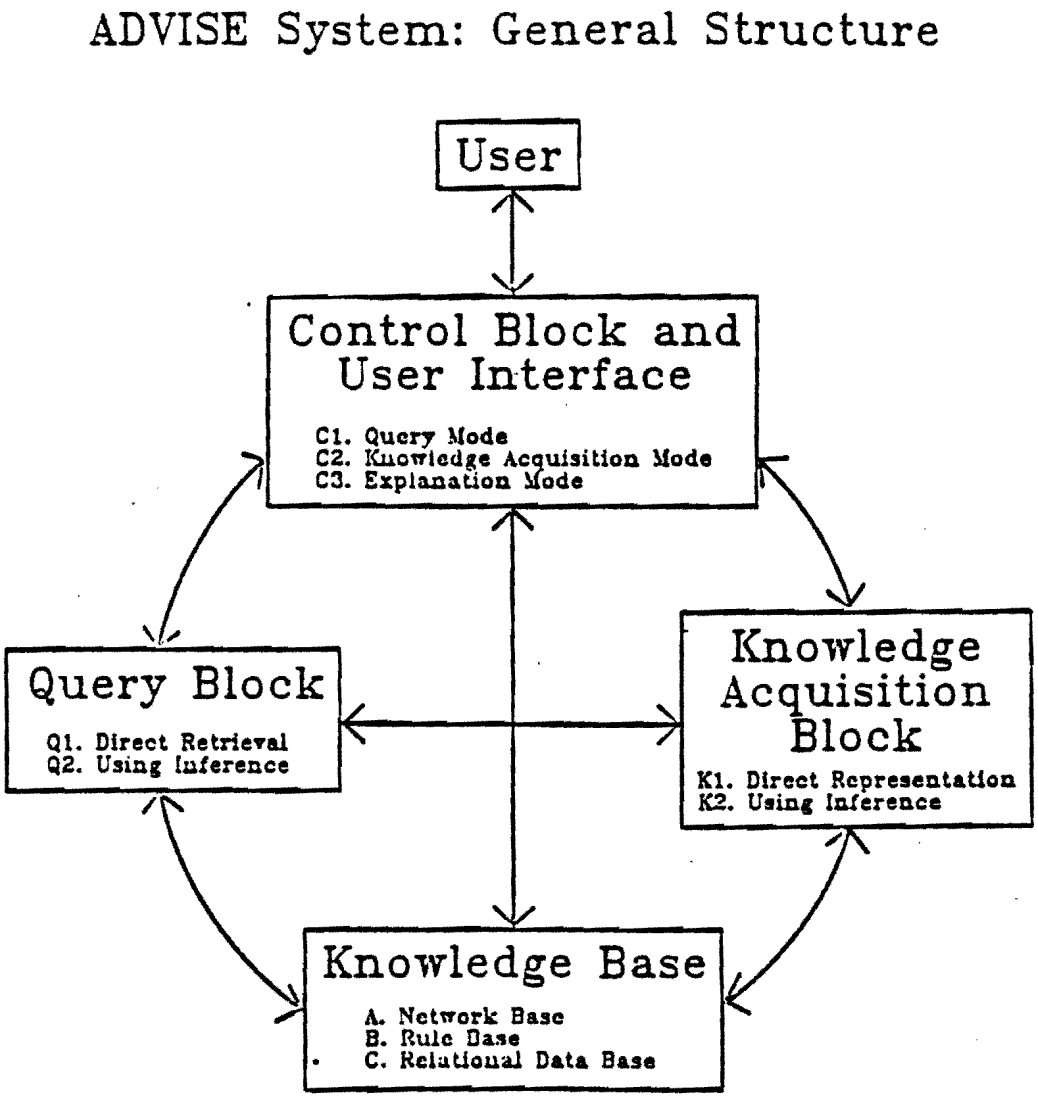


Figure 1. A conceptual level block diagram of ADVISE.

2.2.1. Control Block and User Interface

This component handles three distinct modes of operation. Each mode is listed below with a brief summary of its function.

- *Query mode.* In this mode the system can select questions to ask the user. The system then accepts the user's answers and conducts an inference process which involves the knowledge base and information provided by the user. This allows the system to compute advice with an associated strength of supporting evidence.
- *Knowledge acquisition mode.* This mode coordinates the process of encoding expert derived rules into the knowledge base and the interactive invocation of the separate inductive learning programs. This mode handles specific components designed to define expert rules, refine the rules manually, induce rules from examples, and correct or improve rules with machine aid. This mode allows the system to test rules in interactive mode on individual cases, as well as in batch mode on a collection of cases.
- *Explanation mode.* This mode paraphrases decision rules into English. This enables the user to understand the organization and operation of the system in both query and knowledge acquisition modes. This mode allows the user to interrogate the contents of the knowledge base to determine the steps which led to given advice.

2.2.2. Knowledge Base

The knowledge base consists of three parts, each using a different form of knowledge representation. Each part is briefly described below:

- *Network base.* The network base contains network structures representing general domain knowledge concerning the interrelationship among various conceptual units. The network organization is a form of the *logic net* formalism described in [Baskin 1980].

- *Rule base.* The rule base contains rules in the basic form:

CONDITION ::> CONCLUSION : α,β

where

CONDITION is a formal expression in GVL_1 variable-valued logic system [Michalski 1980a], [Michalski & Chilausky 1980b], [Chilausky 1979], [Michalski et al. 1982], [Michalski et al. 1983c] which involves elementary conditional statements (called *selectors*), linked by various logic operators (including quantifiers). (See Section 3.4.)

CONCLUSION defines the decision or action which is to be taken when the **CONDITION** is satisfied by a given situation.

α is the strength of evidence which supports **CONCLUSION** when the **CONDITION** is completely satisfied ($0 \leq \alpha \leq 1$).

β is the strength of evidence which supports the negation of **CONCLUSION** when the **CONDITION** is not satisfied ($0 \leq \beta \leq 1$).

The rule above is read: CONDITION implies CONCLUSION with forward strength α and backward strength β. This rule is equivalent to the following group of rules:

CONDITION ::> CONCLUSION α
not CONCLUSION ::> not CONDITION α
CONCLUSION ::> CONDITION β
not CONDITION ::> not CONCLUSION β

By providing both α and β for each rule it is possible to use rules for inference in both directions. Below is a rule that illustrates the syntax of GVL_1:

RAIN-RULE

0.8 [BACKGROUND-NOISE = RAIN-LIKE]
[FORECAST(TODAY,TV23-WEATHERMAN) = RAIN]

+

0.2 [NATURAL-LIGHT = LITTLE..MODERATE]
([LIGHTNING = YES] => [THUNDER = YES])
([SEASON = SPRING:0.5 OR SUMMER OR FALL:0.5]
=> [BIRD-NOISE = NONE]) ::>

[RAINING = YES] (0.9,0.5)

The condition of this rule consists of what is termed a *linear module.* The linear module above is in the form: $q_1C_1 + q_2C_2$ The first term that is added above indicates the significant evidence for it raining outside, and the second term is the confirmatory evidence.

The q_1 of the first term, 0.8, indicates that the two conjoined selectors that follow, [BACKGROUND-NOISE = RAIN-LIKE] and [FORECAST(TODAY,TV23-WEATHERMAN = RAIN] are the major pieces of evidence for it raining outside. Furthermore, these pieces of evidence have to occur together. Note that in the second selector of the first term, a function appears: FORECAST(TODAY,TV23-WEATHERMAN). This function could be implemented as a table, algorithmically, or as a rule group.

The second term lends support to the conclusion that it is raining, but its weighting, 0.2, is not high enough to fire the rule if the significant evidence in the first term is not present. This term consists of a selector conjoined with two *implicative* statements. An implicative statement enables one to encode the fact that some conditions must occur for other conditions to lend evidence to the conclusion. Thus for the first implicative statement, if the presence of lightning to be effective there must also be thunder. This encodes the heuristic that lightning from a long distance with no thunder is not evidence for it raining locally. The first selector of the second implicative statement, [SEASON = SPRING:0.5 OR SUMMER OR FALL:0.5] contains *internal disjunction* among three domain elements of SEASON. Two of these domain elements, SPRING and FALL, are weighted so that if the selector is satisfied with one of these domain elements, the selector will return a weeker truth value than if the season was Summer.

The forward strength of evidence for this rule is 0.9, and the backward strength of evidence is 0.5. The backward strength is used when we know it is raining outside and we want to infer what the associated conditions are.

The following computer-derived rule from PLANT/ds illustrates the use of *external disjunction:*

[PLANT_STAND = LESS_THAN_NORMAL]
[PRECIPITATION = NORMAL OR ABOVE_NORMAL]
[TEMPERATURE = BELOW_NORMAL OR NORMAL]
[PLANT_HEIGHT = ABNORMAL]
[CONDITION_OF_LEAVES = ABNORMAL]
[LEAF_MALFORMATION = ABSENT]
[CONDITION_OF_STEM = ABNORMAL]

V

[TIME_OF_OCCURRENCE = APRIL OR MAY OR JUNE OR JULY OR AUGUST]
[PLANT_STAND = LESS_THAN_NORMAL]
[DAMAGED_AREA = GROUPS_OF_PLANTS_IN_LOW_AREAS]
[PLANT_HEIGHT = ABNORMAL]
[CONDITION_OF_LEAVES = ABNORMAL]
[CONDITION_OF_STEM = ABNORMAL]
[EXTERNAL_DECAY_OF_STEM = ABSENT OR WATERY_AND_SOFT] ::>

[SOYBEAN_DISEASE = PHYTOPHTHORA_ROT];

- *Relational data base.* This base contains relational tables which represent any factual information, e.g., examples of experts' past decisions.

2.2.3. Query Block

This component supports queries which are executed by direct retrieval from the knowledge base. This component also supports queries that require inference.

- *Query block using direct retrieval.* This type of query displays the contents of the knowledge base, the network base, the rule base, and the relational data base.
- *Query block using inference.* Computing the most plausible advice for a specific situation is a major function of this system. Queries involving deductive and/or inductive inference are supported for each form of knowledge stored in the knowledge base.

Queries using inference for the network involve *path finding* within the network, e.g., *climbing the generalization tree.* Inference over the network also occurs whenever hierarchal information in the network is used to answer questions about relationships between concepts.

The most important method of providing advice is by doing inference with rules. For a given situation, rules are evaluated by using a method of propagating uncertainties called an *evaluation scheme.* The *control scheme* decides which rule to choose for evaluation. ADVISE offers a host of problem solving

strategies. Local problem solving within a rule is defined by a choice of an *evaluation scheme* (of which several are defined), and particular global problem solving behavior in and among groups of rules is governed by a chosen *control scheme* (of which several are also defined). In ADVISE, there is continual development in local and global problem solving strategies. A key issue to tackle is the the separation of strategic and tactical knowledge in rule base systems. It is felt that there is no adequate strategy specification language, and research into a proper language is under way. We also hope to design a global problem solving language to specify *control schemes.*

Various arithmetic or other transformations of the data items as well as the traditional relational table operations such as *project, select,* and *join* are used in queries on the relational data base. Queries are posed in a modified relational algebra using certain constructs from GVL_1 [Schubert 1977].

2.2.4. Knowledge Acquisition Block

The system design includes knowledge acquisition for each type of knowledge stored in the knowledge base.

- *Knowledge acquisition by direct representation.* The knowledge acquisition block supports knowledge acquisition by direct representation of knowledge provided by human experts.
- *Knowledge acquisition using inference.* Reducing the burden on human experts was intended when including machine based inference as part of the knowledge acquisition process. By defining inference procedures over each component of the knowledge base, the system can now help human experts present a complete, concise, and error free knowledge base.

2.3. A Technical View of ADVISE

Figure 2 is the technical level block diagram of ADVISE that will be discussed in this section. The following section will discuss the current implementation of ADVISE on a VAX-780 under Berkeley Unix. The current implementation does not include all the modules pictured in Figure 2. For a detailed description of the implementation see [Michalski et al. 1983b].

The ADVISE System

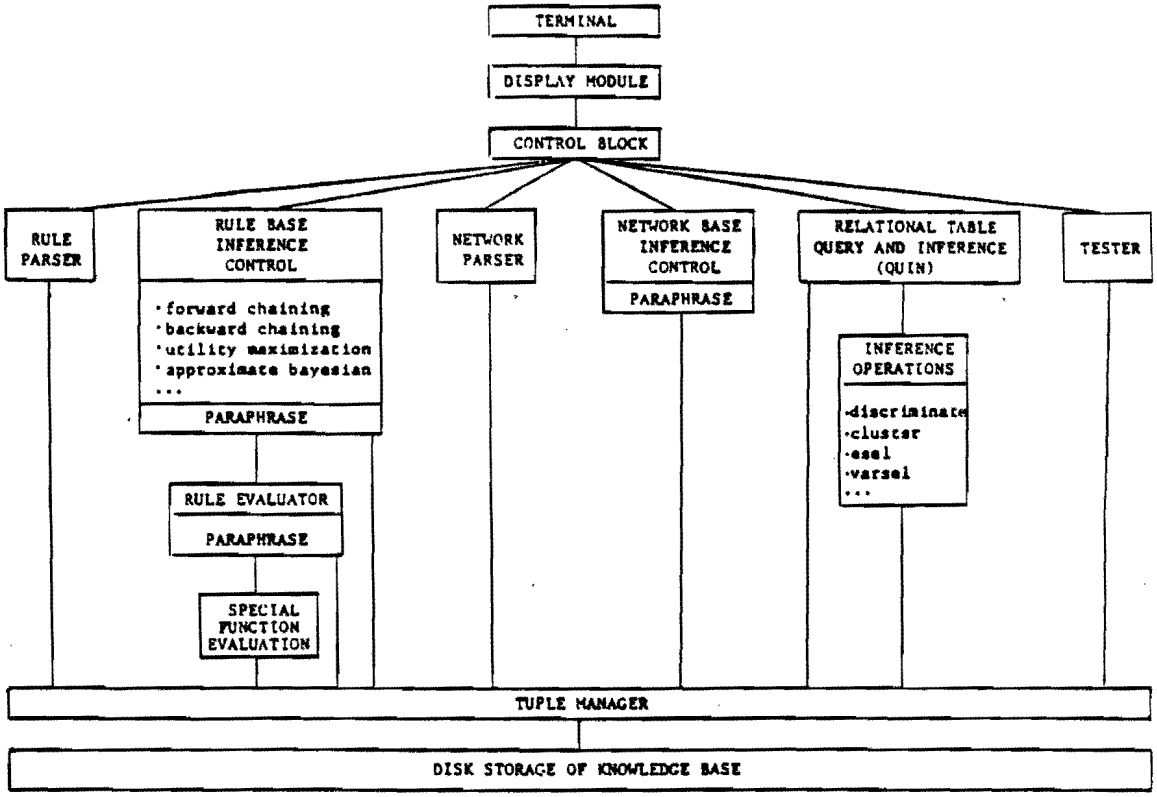


Figure 2. A technical level block diagram of ADVISE.

(1) The DISPLAY module.

All interaction with the user goes through the DISPLAY module. This module provides the user with a good means of man-machine interaction via a touch panel. It logically splits the screen into windows which can have touch-sensitive areas and graphics. Several views of the ORION plasma panel terminal with a touch panel that is used with the display module is shown in Figure 3. The DISPLAY module also supports conventional cursor addressable terminals.

(2) The CONTROL block.

This dispatches the major functions of ADVISE: parsing rules or networks using the parser modules, running a consultation using the rule base or network base modules, relational table query and inference using the QUIN module, and testing parts of ADVISE using the TESTER module. This module is not implemented in the current version of ADVISE.

(3) The RULE PARSER module.

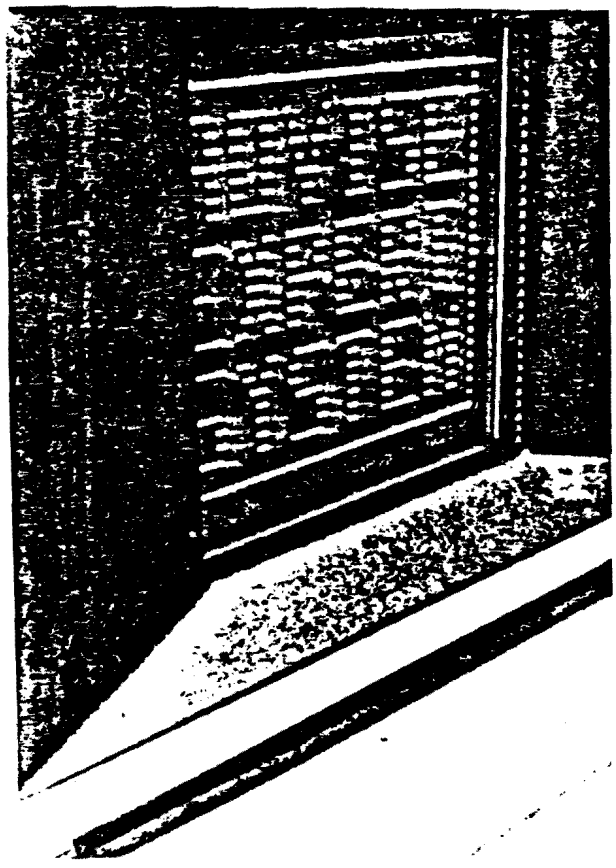
This module parses an extended form of GVL_1 It parses these rules into a parse tree that is represented by a network built by the TUPLE MANAGER module. Currently the RULE PARSER runs in batch mode and is not connected to the CONTROL MODULE or the DISPLAY module.

(4) The RULE BASE INFERENCE CONTROL module.

This module implements a set of schemes (also referred to as control schemes) that use rules to conduct inference. One inference scheme is a backward chaining (with limited forward chaining) control scheme that is used to implement PLANT/cd. Another inference scheme uses utility maximization for variable selection and is used for single-layered rule networks. This inference scheme is for PLANT/ds which diagnoses soybean diseases [Michalski et al. 1982], [Michalski et al. 1983c]. This inference scheme is tailored for machine-derived rules generated by a program such as AQ11 [Michalski & Larson 1978]. Finally, another inference scheme is being built that implements an approximate Bayesian inference mechanism.

(5) The NETWORK PARSER.

This module is used to parse networks. It is currently not implemented.



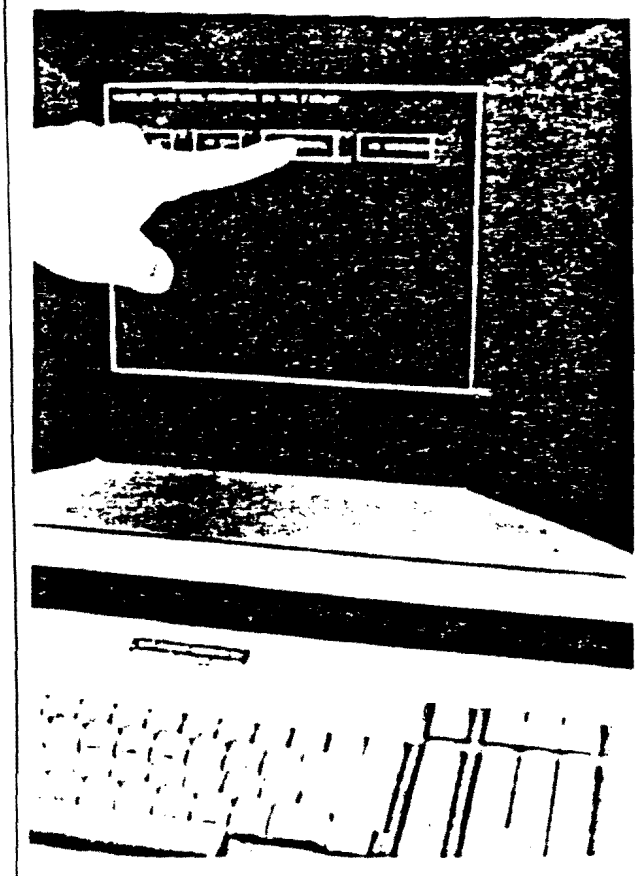


Figure 3. Three views of the ORION plasma panel terminal. The upper view shows what the terminal looks like. The lower left picture is a side view and the infrared emitters that make up the touch panel can be seen. The lower right picture illustrates the user interacting with a *touch target* which is produced using the DISPLAY module.

(6) The NETWORK BASE INFERENCE CONTROL module.

This module is used to carry on inference on networks. This module is currently not implemented.

(7) The RELATIONAL TABLE QUERY AND INFERENCE module (QUIN).

This module is used to do queries and inference on relational tables. QUIN [Schubert 1977], [Spackman 1983] calls a host of data analysis and learning modules such as AQ11 [Michalski & Larson 1978], ESEL [Michalski & Larson 1978], CLUSTER [Stepp 1980], PROMISE [Baim 1982], [Baim 1983], and CONVART [Davis 1981].

(8) The TESTER module.

This module is used to manipulate the tuple network directly. It is also used to exercise other modules in the system.

(9) The PARAPHRASE modules.

These modules are responsible for explaining to the user how rule and network inference is being used during a consultation. Another type of PARAPHRASE module is used to "unparse" a rule from is parse tree form to the human readable form. This module is also used to explain the evaluation of particular rules to the user. Currently the rule and network inference PARAPHRASE modules are not implemented, and only a preliminary version of the rule evaluator PARAPHRASE module exists.

(10) The RULE EVALUATOR module.

This module is responsible for evaluating the premise part of a rule and asserting its consequent if it fires. It evaluates and asserts rule parts under a variety of semantics for logical connectives in a multi-valued logic interpretation.

(11) The SPECIAL FUNCTION EVALUATION (TRAP) module.

This module, also known as the TRAP module, is used to evaluate special functions called TRAP functions that are called within rules. These functions can do such things as special displays, or start up sensors, run models or simulations, etc. Presently, there is one version of this module for PLANT/cd.

(12) The TUPLE MANAGER module.

The basic structure for storing information in ADVISE is the tuple. (Information is also stored in Pascal local variables, but this type of data is particular to the local environment and is not meant to be preserved.) A tuple is a list of memory addresses of nodes. The list length can vary from tuple to tuple (provided it is not greater than some predefined maximum). A node is like a LISP atom and consists of a print name and memory address and has a list of tuples. The node itself is conceptualized to be the head element of each of the tuples on the list of tuples a node has. Tuples are accessed much like LISP property lists are; by *context*. The TUPLE MANAGER is the piece of software that manages a virtual memory for tuples. The TUPLE MANAGER is based on the work in [Baskin 1980]. Figure 4 contains some examples of tuples and illustrates how they can be used to represent a semantic network.

Example of Tuple Representation

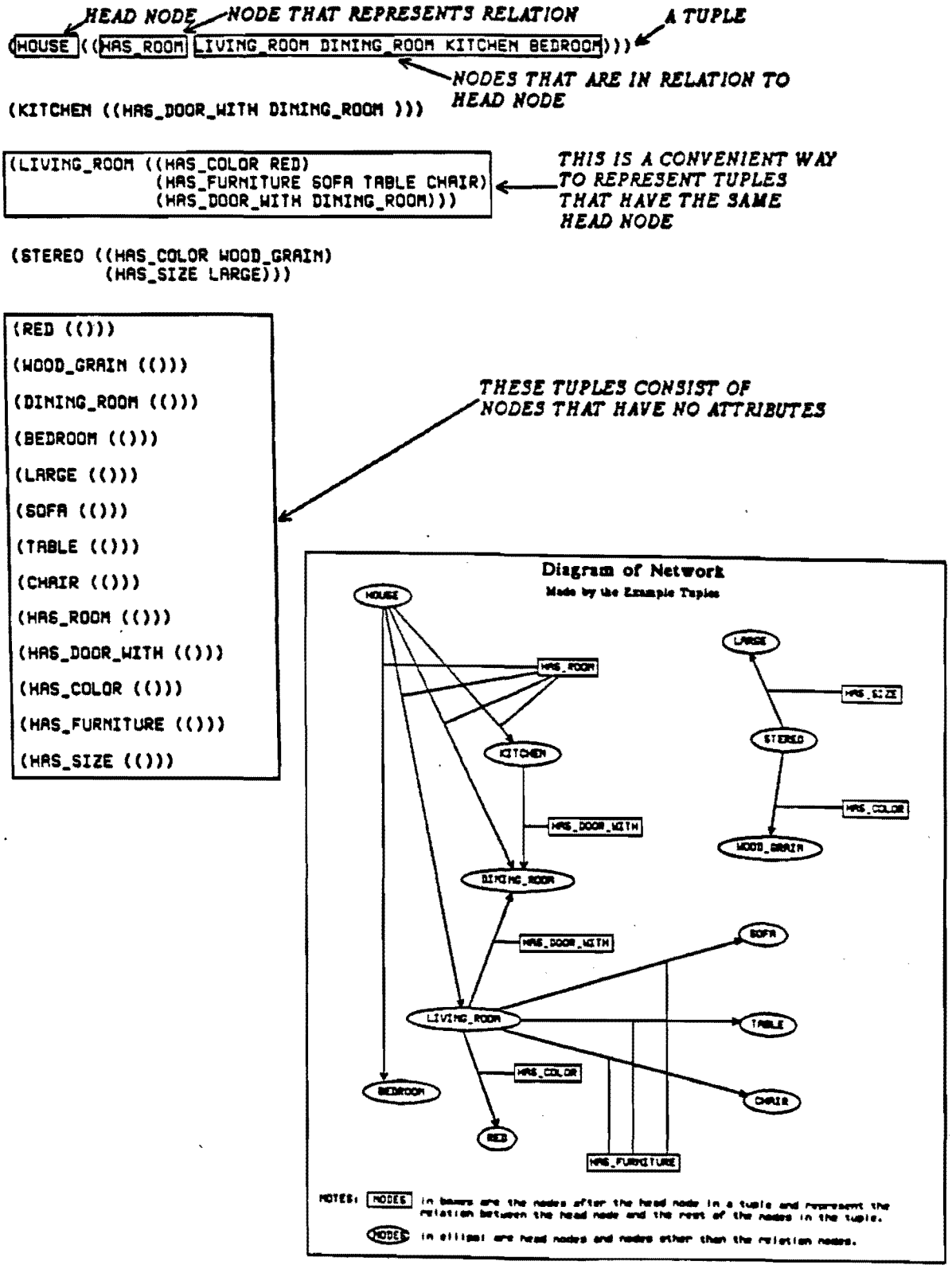


Figure 4. This figure shows an exemplary use of tuples for a semantic net with labeled arcs.

CHAPTER 3

THE BLACK CUTWORM DAMAGE KNOWLEDGE BASE

3.1. The Problem Domain

The Black Cutworm (BCW) is a sporadic corn pest that in any given year damages 2 to 10% of the Midwest corn acreage above the economic injury level. It damages corn by cutting it around the base of the stalk. The following is a summary of how experts view the process in which a cutworm population damages a field. A process graph is provided in Figure 5. This graph is a perspective graph of the damage process. The *horizontal axis* represents the number of Centigrade degree-days (CDD)[1], which is a measure of time in terms of the total heat input to the system. As can be seen on the graph, later months provide much more heat than the early ones. The *vertical axis* represents the level of the cutworm population and the corn population. Population level of the corn represents the number of live corn plants and is unrelated to corn height. For Illinois, there are typically three generations of cutworms. Usually, the first generation does the corn damage. In this area (Central and Northern Illinois), Black Cutworms do not *overwinter*. The *depth axis* represents the different groups involved in the damage play. They are corn, eggs, and the seven instars of the Black Cutworm. Instars are larvae between molts. First instar larvae are between the hatch stage and the first molt. There are seven instars in the life of a Black Cutworm. Each instar represents successively more mature Black Cutworm larvae. The reader is referred to [Sherrod et al. 1979] for more details of Black Cutworm damage in Illinois.

---

[1]Degree-day is defined as follows:

$$dd(t) = \int_{t_0}^{t} F(\alpha)d\alpha\,, \qquad F(\alpha) = \begin{cases} T(\alpha) - \beta & \text{if } T(\alpha) > \beta \\ 0 & \text{if } T(\alpha) \leq \beta \end{cases}$$

where $T(\alpha)$ is the temperature as a function of time, and β is the threshold temperature at which both corn and cutworm development commence and is assumed to be 10.5 °C. A Centigrade degree-day is based on the Centigrade scale.

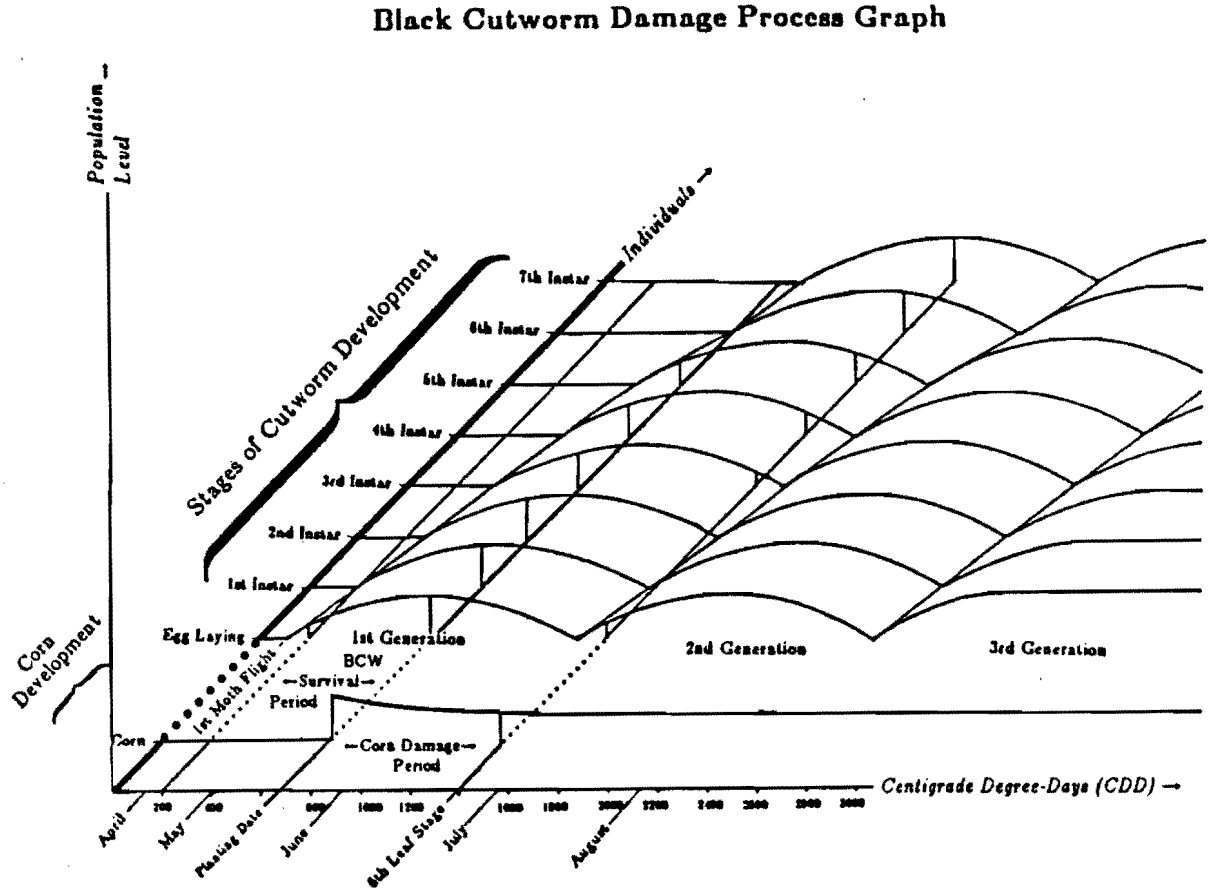


Figure 5. This perspective graph depicts the damage process. Damage occurs when there are larvae in the third instar or older and corn that is at a leaf stage less than 6.

The Black Cutworm season begins in Illinois when Black Cutworm moths are blown in from the south. This typically occurs around the middle of April. Egg carrying female moths lay a fixed number of eggs during the season. The number of eggs laid in the field is a function of how attractive the field is to the moths. It is believed that the predominate predictor of field attractiveness is weediness of the field.

During the period of egg laying, the field will develop an egg population, the dispersion of which is through time. This is represented by the curve labeled "Egg Laying" in Figure 5. This population will develop into a larval population that could eventually cause damage to the corn crop. The larval population in Figure 5 is represented by the curves labeled "1st Instar" through "7th Instar."

Not all the larvae will survive. If food sources are taken away (e.g., by tillage), then this will reshape the original larval population. This reshaping will affect both the amplitude and width of the population curve. This is called *larval survival*. The larval survival period, shown in Figure 5, is between the emergence of the first instar larvae and corn emergence.

Damage begins during corn emergence. The corn damageability is a function of both corn maturity and larval maturity. The more mature the larvae, the more damage they can do. If the corn is planted early in the season, the first (as well as the rest of the BCW population) will be unable to cause any damage to the corn. The period of damage lasts until the youngest of the BCW population reaches the pupation stage or the corn is too big for the cutworms. Corn is mature enough by the seventh leaf emergence. Thus, the damage period in figure 5 is between the first and sixth leaf stage.

3.2. Deriving the Expert Rules

One of the first things mentioned by experts that were interviewed was that BCW damage is a relatively rare event. It is therefore difficult to get a large enough data set to cover all the variability in the input descriptors (variables). Furthermore, the space generated by these descriptors is very large. What is available are some incomplete data for the years 1976, 1977, and 1978, somewhat better data for the year 1980, and fair data for the year 1982. Moth flight date, a descriptor thought to be very important, is only available for a few stations and only for some of the years. Moth flight data was extensively collected in 1982, however.

Since BCW damage is a relatively rare event, the people familiar with it have a weak "seat of the pants" feel for predicting the amount of damage from what is observable. It was therefore a very difficult task to obtain what knowledge there was on this subject and express that knowledge in the form of rules. The best that could be done was to use the simulation programs that one of the experts, Steve Troester, has written [Troester et al. 1982a,b,c,d], [Troester et al. 1983]. Troester has implemented a systems dynamics model which was developed in collaboration with others to simulate the dynamics of the process, which can be "fine tuned" by trial and error.

Also available are the results of John Davis' masters thesis [Davis 1981]. He ran the 1980 data through his program, CONVART. His program created new descriptors of time-varying data based on mathematical transformations of initial descriptors characterizing the field and BCW damage. The output of CONVART was then used as input to the AQ11 inductive learning program which generated a set of rules that described the data. Unfortunately, the derived rules only categorized damage into two classes: *damage* and *no damage*. Also his rules are only good for the year 1980, because they were derived from data collected solely in this year. To derive general rules, such descriptors as moth count history and weediness history (that extends beyond spring tillage practice) are needed. The above two descriptors are also key factors in the prediction of extent of damage as well.

Because of the problems with the machine derived rules and with expert opinion, it was decided to base PLANT/cd on Steve Troester's work. However, the approach taken by Davis could lead to better results and this has been indicated by the work of Paul Baim [Baim 1982], [Baim 1983]. The major problem with this approach has been the lack of complete damage data. Perhaps with better data, better results could be obtained.

3.3. The Implementation within the ADVISE System

In order to build an expert system that allowed easy modification and upgrading of encoded knowledge and had the eventual capability of explaining its reasoning, a good source of expert knowledge was needed. The best available source of knowledge was Troester's *deep* model of the dynamics of BCW damage. The attempt to derive a set of rules that predicts extent of damage from expert opinion failed. Opinion existed, but it was not even across the whole problem domain. For instance, no one seemed to have good opinion on the synchrony problem (i.e., if a person was given moth flight dates and planting dates, he would not have a good feel as to how well in synchronization corn development and cutworm development are), nor a good feel of corn development, cutworm development, or survival. The best opinion was in the area of field attractiveness, as well as *exceptional cases* like flooding. This was not surprising since all those involved claimed that they really did not understand the whole problem. The reasons for this lack of understanding have been explicated earlier; primarily because BCW damage is a

comparatively rare event and not enough knowledge of the subject has been accumulated. Also, because of synchrony, the total number of situations is vast. This leads to the worst possible situation one could think of: a vast description space, and few examples to fill it.

After realizing that Troester's deep model was the best source of information, the following question was faced: "How does one take the knowledge contained within the deep model and place this knowledge in production rules?" The way that seems best is to break the deep model into subprocesses and make the subprocesses functions that are invoked in the right-hand side (RHS) of rules. Thus the coordination of these subprocesses would be handled by the control scheme as well as the logic in the rules. Figure 6 is a dependency graph that illustrates what these subprocesses are. User inputs to these subprocesses (denoted by the thin-lined boxes in Figure 6) are:

(1) Moth trap catches for the field or the surrounding area. These are taken on a weekly basis for 11 weeks beginning about the time first moth flight occurs. The model was designed for moderate infestation and the upper range for moth trap count on a weekly basis is 300. A typical range would be 0-60 moths. Observation dates range from March 1 to July 1st.

(2) A measure of the weediness of the field during the time moth trap catches are being recorded. This is in the form of an *attractiveness rating* for the 11 weeks that trap counts are taken. It is given as a number which ranges from 0.0 to 1.0.

(3) The larval age spectrum can also be input to the model. This data, in the form of larval counts in instar categories, is provided to the model when damage is occurring. These counts range from 0-15 per 100 feet-of-row, and an absolute upper value would be 30 per 100 feet-of-row.

(4) Whether flooding in the field occurred.

(5) Location of the field and observation date. The location allows the system to associate a weather station where temperature data will be taken. The model can be run for several of the states near Illinois. They include Indiana, Missouri, and Iowa. The observation date is used to build up the temperature data used by the model (denoted by the double-lined box in the figure). It is also used by the model as a time reference from which projections into the future will be made.

Black Cutworm Damage Dependency Network

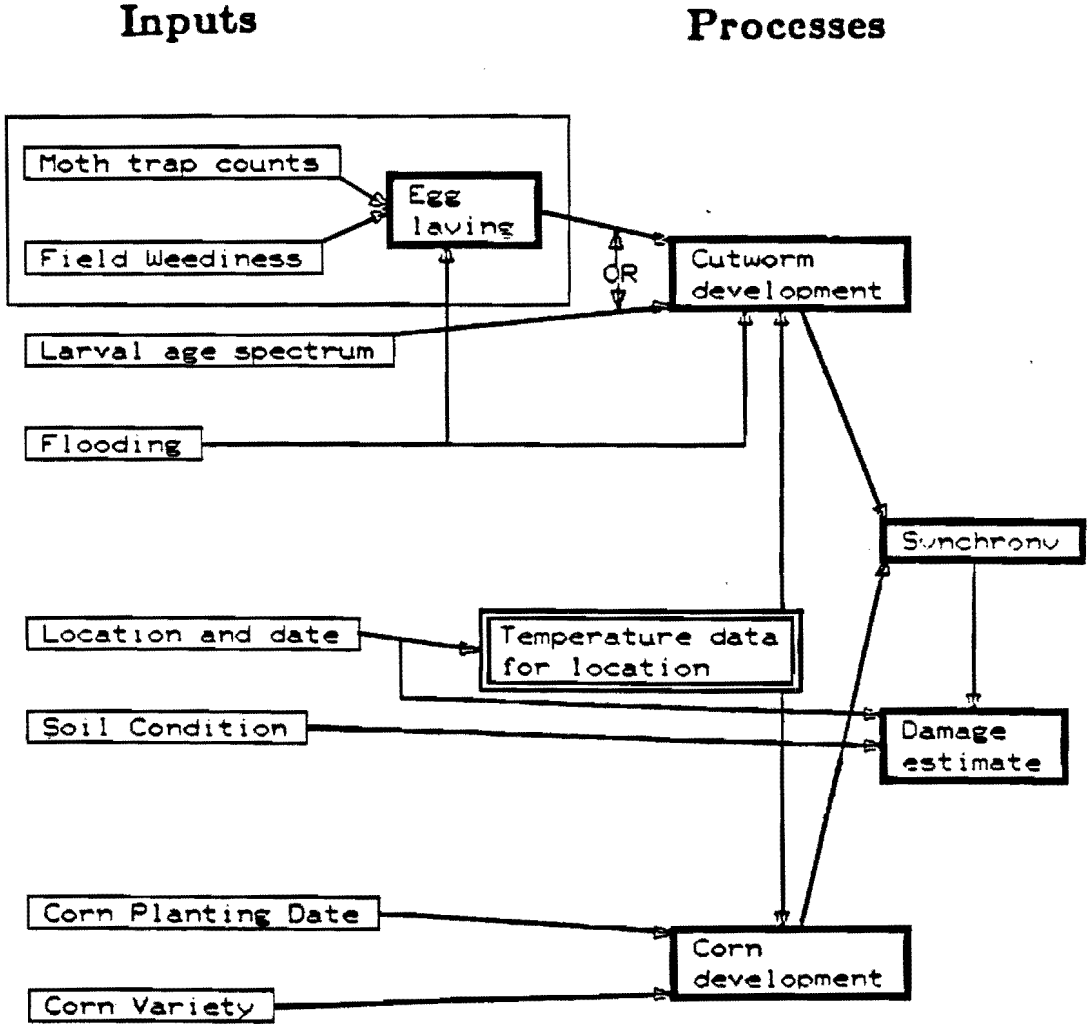


Figure 6. A dependency network of the processes that are modeled in PLANT/cd. The boxes with thin borders represent inputs. The boxes with thick borders represent subprocesses. The double-lined box entitled "Temperature data for location" represents a data base used in the model.

(6) Soil condition. The model uses the values *normal, wet,* and *dry.*

(7) Planting date of the corn. Typical planting dates range from April 15 to June 10. Southern areas may have dates that are in late March. The absolute range is from March 20 to July 1.

(8) Corn variety. This is needed because fullseason and midseason corn have different rates of development. The model distinguishes between *fullseason* and *midseason* corn.

Below is a short description of the interconnectivity of the dependency network based on the work of Troester [Troester et al. 1982a,b,c,d], [Troester et al. 1983]. Troester himself broke his simulation programs into several of these subprocesses.

(1) *Egg Laying.* This is the process of getting an egg population laid in the farmer's field. This is predicted from the moth trap counts and field attractiveness. Also influencing this process is flooding which makes the field very attractive to oviposition.

(2) *Cutworm development.* This is the process of taking the egg population in the field over time and developing it into a larval age spectrum. It includes the survival of the cutworm which is influenced by food supply. (Food supply is derived from the the measure of field weediness.) Cutworm development is influenced by the temperature, and temperature data from the closest weather station is used. There can be two inputs to cutworm development; one from the egg laying process described above or actual larval counts taken from the field can be supplied by the user.

(3) *Corn Development.* This is the process of the corn developing. This is influenced by the variety of corn planted, the planting date (possibly estimated), and the temperature data. If the farmer provides moth trap data (before planting date), then he can vary the planting date of the corn to minimize his losses if given good enough estimates of his losses due to BCW damage.

(4) *Synchrony.* In order to estimate BCW damage, the age spectrum of Black Cutworm larvae has to be known at different periods of corn development. Synchrony is the process of putting BCW development in terms of corn development.

(5) *Damage Estimate.* This is the process of summing the damage done to the field across time and estimating the total damage done to the field. This takes into account the fact that some of the

damaged plants will recover. Total amount of damage is influenced by the soil condition, corn variety, and several other characteristics of the geographic area that the field is in. One example of such a characteristic is the parasitism rate. This is the mortality rate of the larvae due to parasites. In Troester's work the parasitism rate can vary from state to state.

Troester's programs are broken into two cases. One case is when damage is to be predicted in the future and there are no larvae in the field [Troester et al. 1982c,d]. This case uses moth trappings to predict what the population will be in the future. The other case requires that larval counts be entered. Of course, the later situation yields a more accurate prediction. Troester was not certain with the simulation results in the first case. He has proposed to "fine tune" this case.

Just as Troester's programs distinguish two cases, the rules distinguish the same two cases. A backward chaining control scheme has been identified as a good control scheme for the type of rules that has been generated. Troester's simulation programs, originally coded in FORTRAN, have been rewritten as PASCAL procedures that are callable by rules. The module that they are in is called the TRAP module. The PLANT/cd TRAP module will be described in Section 5.2.

Below is the English form of an example rule from PLANT/cd:

RULE12

[This rule is tried in order to find out about Black Cutworm development]

If: 1) The degree-day table is known,
2) The observation date < planting date, and
3) The egg population in the field is known.

Then: Simulate BCW development and assign the results to the variable BCWD and display results of the simulation.

This rule is formulated in PLANT/cd as:

RULE12

[DD = KNOWN][OBDATE < PLANTDATE][FEGGS = KNOWN] ::>
[BCWD = TRAP(15,OBDATE,PLANTDATE,FEGGS,DD)]TRAP(9,OBDATE)
!SIMULATE BCW DEVELOPMENT AND DISPLAY RESULTS

where

DD denotes degree-day table,

OBDATE denotes observation date,

PLANTDATE denotes planting date,

FEGGS denotes the egg population in the field,

BCWD denotes the results of BCW development,

TRAP(15,OBDATE,PLANTDATE,FEGGS,DD) is a function to simulate BCW development and is used to assign a value to **BCWD**,

TRAP(9,OBDATE) is a function that displays the results of BCW development and is treated as a predicate by the ADVISE RULE PARSER, and

15 and **9** indicate what *TRAP* functions to call. (See below and Section 5.2.)

This rule uses an extension of GVL_1 to allow functions in the reference of a selector and to replace a selector. (See next section.) The concatenation of predicates represents conjunction.

These rules make extensive use of functions that allow escapement into Pascal code. These functions are termed TRAP functions (from the notion of a trap door which allows one to escape) and in this knowledge base provide a connection between knowledge encoded as rules and knowledge encoded algorithmically. The last line of the rule is a comment to help explain the purpose of the rule. Comments begin with the "!" character.

In order to provide meaning to the variables used in PLANT/cd, they have to be *declared*. Below is an example declaration of the VARIETY variable used in PLANT/cd:

```
VARIETY NOMINAL = (FULLSEASON MIDSEASON) ← Defines the domain of the variable.
PROP = PROMPT
WHAT TYPE OF CORN IS BEING PLANTED?      ← The prompt used to query for the
%%                                         value of the variable.
PROP = TRANS
THE CORN VARIETY                         ← What the variable means
%%                                         in English.
PROP = LABDATA                           ← This variable is a LABDATA variable.
T                                          (See Section 4.1.)
%%
PROP = NOUNKNOWN                         ← UNKNOWN is not accepted as a
T                                          value for this variable.
%%
;
```

In above, the variable VARIETY is defined to be a *NOMINAL* type variable, which means that the domain elements of VARIETY, FULLSEASON and MIDSEASON, have no ordering. The keyword *INTERVAL* is used to indicate a variable that has an ordering among its domain elements. The keyword *STRUCTURED* is used to indicate a variable that has a hierarchical relationship among its domain elements, but this is not supported by the PLANT/cd control scheme. See [Michalski et al. 1982] and [Michalski et al. 1983c] for more details of domain specification. Properties or attributes of a variable are given by the lines that begin with **PROP** =*property* and the text on the following lines, ending with the %%. For instance, the prompt to use when asking for the value of VARIETY is "WHAT TYPE OF CORN IS BEING PLANTED?" and this text is placed under the PROMPT property of VARIETY. Some properties are "yes-no" in nature and the text of these properties is simply "T". An example of this is the LABDATA property. It is sufficient to know that the variable VARIETY has the LABDATA property and so the text of the property is the token "T". For specific syntax used for rule and variable declarations, see [Michalski et al. 1983b].

3.4. The Extended GVL_1 Used in PLANT/cd

This section will present the extensions and subset of VL_1 that is used in PLANT/cd. A subset of VL_1 was used because the rules did not require the full power of VL_1. For instance, the PLANT/cd rules did not use disjunction but this is allowed in VL_1. However disjunction and other operators such as implication and equivalence could have been used if needed. For more complete discussions of VL_1 see

[Michalski 1980a], [Michalski & Chilausky 1980b], [Chilausky 1979], [Michalski et al. 1982] and [Michalski et al. 1983c]. See [Michalski et al. 1983b] for a more detailed description of the extended VL_1 used in ADVISE. RULE12 of the last section will be the example.

As explained in Section 2.2.2, the left-hand side (LHS) and the right-hand side (RHS) of RULE12 are formal expressions which involve elementary conditions called *selectors* linked by various variable-valued logic operators. (In the case of PLANT/cd, only conjunction is used.) A selector is a statement that relates a variable to one or more elements from its domain. Selectors are surrounded by square brackets. The reference of a selector is the position on the RHS of the relational operator and represents a subset of the value set of the variable on the left of the relation.

As an extension to VL_1, the meta-value *KNOWN* is allowed as a reference in a selector and the selector is evaluated to be *true* if the value of the variable in the selector is known. Also the meta-value *DEFINED* is allowed as a reference in a selector and the selector is evaluated to be *true* if the value of a variable has been defined. KNOWN is used in the first and third selector of RULE12. The values of variables have an attached *confidence* which reflects the degree of certainty (or belief) that the system or user has in the value of the variable.

There are implicit variable-valued logic conjunctions between the selectors on the LHS of RULE12. The LHS and RHS of rules are evaluated/executed by the RULE EVALUATOR module of ADVISE. The way the variable-valued logic operators are evaluated is selectable from a set of predetermined *evaluation schemes*. The evaluation schemes for conjunction include *min* which is used in PLANT/cd, *prod* (product), and *ave* (average). Rule execution tries to *satisfy* the degree of certainty that is asserted for the RHS. Thus the execution of the RHS causes variables to be assigned values with an associated degree of certainty. In PLANT/cd, certainty factors are not used to any major extent. The α/β strengths of evidence mentioned in Section 2.2.2 are also not used.

TRAP functions are a type of function that are used in the reference place of a selector in PLANT/cd. (See Sections 5.1 and 5.2.) An example of this is the selector in the RHS of RULE12. In the first selector, TRAP(15,OBDATE,PLANTDATE,FEGGS,DD) is used to provide a value for the variable

BCWD. Trap functions can also operate at the level of selectors, and when evaluated they provide a truth value as selectors do. This provides a way to call a function for its side effects only. IN RULE12, TRAP(9,OBDATE), is used in this way to display a table.

Finally, VL_1 has been extended to allow arithmetic expressions as the reference of selectors. An example of this is RULE23 in Appendix A.

3.5. The Deep and Surface Model Connection

As many researchers in the field have realized, knowledge encoded as an algorithm has a place in expert systems. The algorithmic representation captures precise causal relationships. This level of knowledge representation is often called a *deep model* of some particular domain. In contrast, the knowledge represented as a set of inference rules is called a *surface model*. To quote Peter Hart [Hart 1982]:

> By surface systems I mean those having no underlying representation of such fundamental concepts as causality, intent, or basic physical principles; deep systems, by contrast, attempt to represent concepts at this level. At the extremes, a surface system directly associates input states with actions, whereas a deep system makes deductions from a compact collection of fundamental principles.

This distinction is thought to have some validity in explaining how people think. A person usually first learns the deep model and once that is mastered (his world model is extended to include the problem being mastered), he develops a surface model for the domain. The surface model can be described as being *shallow* since it maps inputs of the domain to outputs without too much intervening cognition. Also, the surface model is built from frequently observed situations and is often only an *approximation* of the actual process. This surface model is used for rapid and general evaluation and prediction. When more precise and detailed information is needed, a person resorts to the deep model.

Deep models have been integrated into several expert systems:

- MYCIN generated therapy selection by an algorithmic process [Clancey 1978].
- TAX ADVISOR, a knowledge base for tax advice, used a cash flow analysis program, called EXECPLAN, for filtering recommendations generated by rules [Michaelsen 1982].

- PLANT/cd encodes much of its expert knowledge in the form of simulations which are called from rules.

These deep models are used as *gurus* in the sense that they contain knowledge considered to be correct and precise. Typically this deep knowledge is implemented as a mathematical simulation model.

There has been little work on how the deep and surface model connection should be done. It is instructive to analyze how the three examples above did make the connection. In MYCIN, a record of the possible diseases was kept for the therapy selection process. Therapy selection then made a selection of drugs to use. The therapy selector used a generate-and-test procedure. The therapy selector was evoked as a function from the RHS of the top (goal) rule in MYCIN's inference tree. The therapy selector in turn invoked the control scheme to use rules as a post critic. (It also used rules for some intermediate steps. See [Clancey 1978] for details.)

In the TAX ADVISOR expert system a record of simulation inputs was prepared when a rule fired which placed an item on the list of actions that might maximize one's tax advantage. These inputs included the costs involved in doing a particular action. After the session with the TAX ADVISOR, EXECPLAN was used to select subsets that were accepted by its cash flow analysis from the action list.

In PLANT/cd, the feedback between the surface model and deep model is tighter. The connection between the deep and surface model is made when functions are called in the RHS of rules that simulate a certain subprocess. In this knowledge base, the LHS of the rule serves as a means of making sure all the inputs of the function are known before the rules are fired and the function subsequently called. This method is illustrated in Figure 7.

What is the closest approximation of a deep model function in terms of surface model rules? A finite-valued function can be represented as a table. Each row of such a table represents a mapping of inputs to one output value of the function. This table can also be viewed as a decision table. There is a correspondence between a decision table and a set of rules. Each rule corresponds to a row in the decision table. A set of rules that correspond to a decision table will be termed a *rule group*. (This rule group is a more restricted form of rule groups that can be parsed by the ADVISE rule parser). Each rule provides

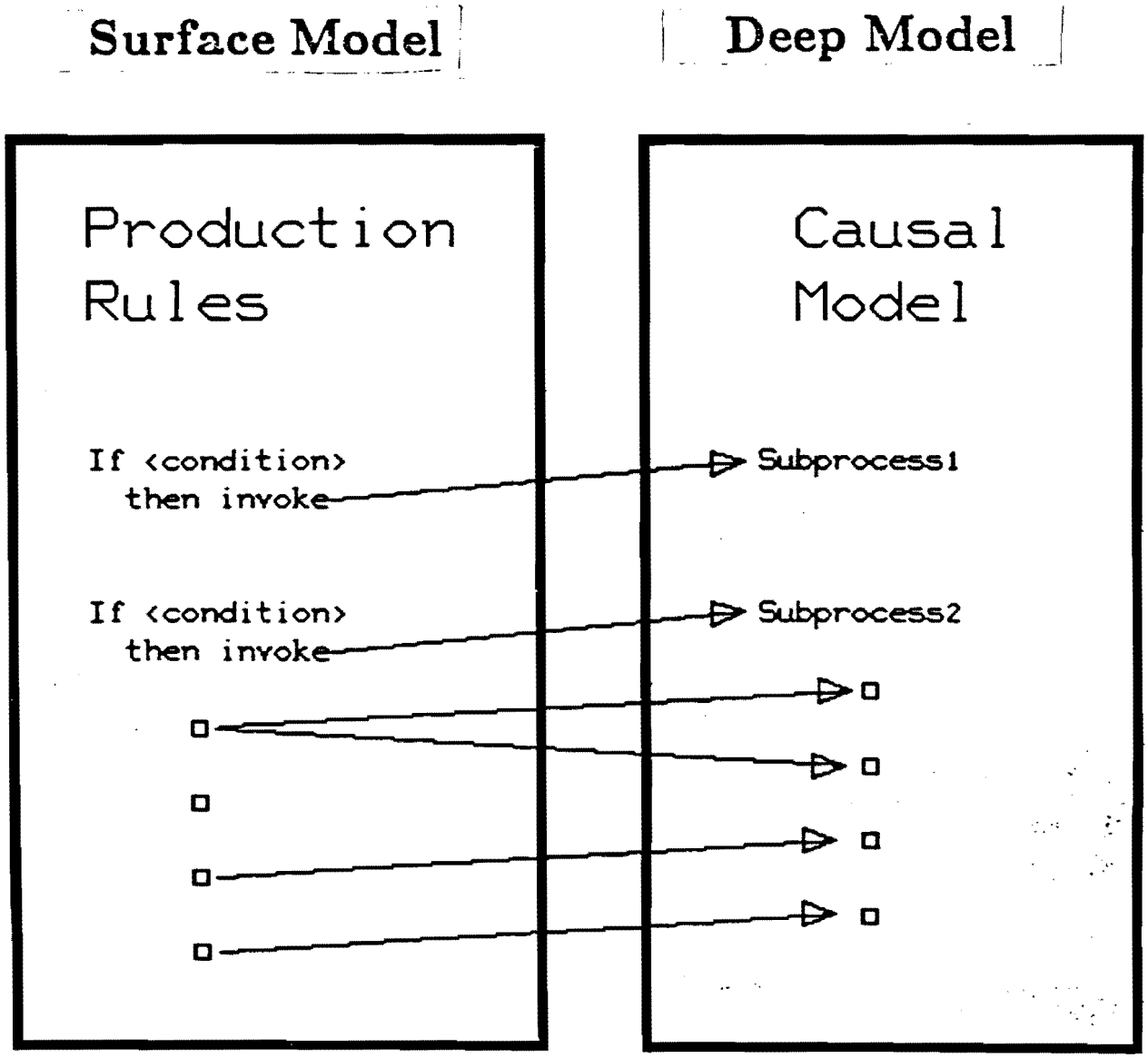


Figure 7. A diagram of the surface and deep model connection in PLANT/cd.

one value for the goal variable of the rule group. For functions whose inputs are continuous-valued or whose outputs are continuous-valued, the values can be discretized and a rule group that approximates the function's behavior found. The process of finding a rule group that approximates a function can be done by the expert or by machine. Machine induction can be used to find a rule group by using the function to be approximated as a case generator. The induction program then uses these cases as input. The domain expert can also provide an approximate description of a function by writing a rule group that corresponds to it.

In a rule base, both the actual function and corresponding rule group can be used. For accurate answers, the function can be used. For fast approximate answers, the rule group can be used. The rule group can also be used for paraphrasing the workings of a function to the user.

It is proposed that a specific function and its corresponding rule group be packaged into a *knowledge packet*. A control scheme could be designed for knowledge packets that would decide to use the rules or functions. In fact, the *maximum utility* control scheme used in PLANT/ds [Chilausky 1979], [Michalski et al. 1982], and [Michalski et al. 1983c] is really a control scheme for evaluating rules *within* a knowledge packet. This control scheme efficiently evaluates a set of rules that has the following properties:

- all the rules form a single-layered inference network, and
- each rule in a group updates a common RHS variable.

A backward chaining mechanism that knows when to invoke the knowledge packet control scheme would be a candidate for the *in between* knowledge packet control scheme. The function in a knowledge packet would be free standing, unlike the functions that are used in PLANT/cd and MYCIN. In these systems, the functions are called within a rule that mentions the function. In a knowledge packet, however, a function represents a set of rules as a whole. The control scheme would have knowledge to invoke the function when constraints like "all the arguments to the function are known" are met. In MYCIN and PLANT/cd these constraints are encoded as *meta*-conditions in the LHS of the rule that contains the function. These meta-conditions act to control the flow of inference in MYCIN and PLANT/cd. When these meta-conditions are mixed with other conditions that express non-control domain knowledge, the

principle of keeping separate *strategic* (control) and *tactical* (non-control) knowledge in rules is violated.

Expert systems that have combined deep and surface models are called *multi-level* systems by Hart [Hart 1982]. Such expert systems have many issues yet to be resolved and some of the issues pointed out by Hart are:

- Identification of the important levels of representation of a given domain.
- Determining the best representation for each of the levels.
- Determining the formal relations that should guide the formation of each of the levels. Should there be a Correspondence Principle that states that each succeeding level is a simplification of previous ones, as Newtonian mechanics is to Quantum mechanics?
- Determining the best use of each of the levels. Each level could be used for purposes of validation, explanation, and control. Levels that encode deep knowledge could be used to *compile* surface models.

The author believes that the use of deep models to enhance the explanation of surface rules is an answer to some of the problems of "shallow" explanations given by systems that encode knowledge at a single level [Clancey 1981]. The ADVISE project is also embracing the concept of compiling surface models from deeper levels of representation. We are using this concept to compile expert systems for microcomputers from a more general one on a development computer (currently a VAX/780).

CHAPTER 4

THE BACKWARD CHAINING CONTROL SCHEME

4.1. User Description

This control scheme was patterned after EMYCIN's [van Melle 1979], [van Melle et al. 1981]. Besides the basic control, it features ANTECEDENT rules (limited forward chaining, see below), LABDATA variables (variables that are marked to be asked before any other variable), and multiple goals for a rule group.

The control scheme makes use of properties. Such properties are indexed pieces of text a rule or variable can be assigned. For examples of how properties are specified, see Section 3.4 and the variable listing of PLANT/cd in Appendix A. The control scheme uses the PROMPT property of a variable to solicit for the variable's value. TRANS property is used to present the variable to the user. The TRANS property of the consultation goal variables are used in displaying the results of a consultation by displaying the TRANS property of the goal variables along with the values of the goal variables. If there are rules that could be used to infer the value of a variable, then the ASKFIRST property is used to indicate that an attempt should be made to obtain the value of the variable from the user before using any rules. ANTECEDENT rules and LABDATA variables are marked by the use of the ANTECEDENT property and the LABDATA property respectively. The text for these properties is simply "T." If a variable is never to have the "UNKNOWN" value provided by the user, then it is marked with the NOUNKNOWN property.

ANTECEDENT rules allow limited forward chaining in a backward chaining mechanism. These rules fire whenever their right-hand sides (RHSs) are satisfied; even if they have not been "back chained" to. This forward chaining is limited to the firing of one rule in a forward reasoning chain per cycle of the control scheme. The algorithm for forward chaining can be paraphrased: "Whenever a variable is queried or is updated by the execution of a RHS, all ANTECEDENT rules that refer to that variable on their

left-hand side (LHS) are evaluated."

GOAL variables are the variables whose value/confidence pairs are displayed to the user at the end of a session. They usually represent the entities on which we seek advice. The control scheme allows more than one GOAL variable.

LABDATA variables represent basic input information and are used throughout the consultation session. LABDATA variables are *found out* (either by asking or inferring) before the GOAL variables. Unlike EMYCIN, in which LABDATA variables are always queried, here these variables can also be inferred. The author's past experience with EMYCIN suggested that such an option is sometimes useful.

There are a number of additions that could be made to the present version of the control scheme:

- Self referencing rules. This type of rule (the same variable appears in the LHS and RHS of the rule) can be used to incrementally update the certainty of a variable's value if more supportive evidence is acquired. The semantics of these rules can be paraphrased roughly: "If you have already decided something (the value for the common variable in the LHS and RHS) and further evidence comes in to support it, then update the confidence of your decision." These rules can also be used to critique previously supplied values.
- EMYCIN contexts and between context reference. Contexts provide a way to eliminate redundancy in a knowledge base. They do this by introducing the notion of a generic organizing object to which the rules refer. For instance, in MYCIN there is a context for cell cultures. The rules in this context refer to the generic object, culture. Without contexts, there would be a separate set of rules for each of the cell cultures a person could have. With contexts, there is one set of rules, and when an instance of the context is made, the rules are used to infer values for that context instance.
- Variable blocks. These are groups of variables that can be queried together. This could be done by setting up an empty table for the user to fill in - as in EMYCIN[1], or by multiple display module windows on the same screen.

[1]PLANT/ds [Michalski et al. 1982] also uses this method.

- A consultation typescript file. This is used to have a hard-copy transaction of the consultation.
- The ability to save and reload prior consultations. This can be used to have a "canned" set of demonstration consultations. It also can be used to test a knowledge base.
- A mode to exercise the knowledge base with test cases. This mode allows a knowledge base builder to test a knowledge base by running test cases in "batch" mode. These cases are stored in a file.
- Help and explanation facilities. At present, this control scheme does not explain its reasoning to the user. There are no help commands to guide the user in the use of the system. These features give expert systems a user friendly environment and should be included.

4.2. Control Scheme Internals

The most important parts of the backward chaining control scheme, embodied in a program called BWARD, are illustrated in flowchart form in Figures 8,9 and 10. The top level of BWARD is shown in Figure 8, and Figures 9 and 10 contain the flow charts for two important procedures of BWARD: FINDOUT and INFER. In these figures the names of all procedures are in capital letters. Some of the boxes in the flowcharts have a line that separates the name of a procedure from what that procedure does. The procedure names are simplified a bit since the actual names start with the module two letter prefix "BC" to abide with ADVISE module naming conventions. Also many of the low level procedures (not shown in the flowcharts) are in a set of utility procedures, called CSTOOLS, that is shared between PLANT/cd and PLANT/ds. These procedures are prefixed with "CS."

The top level of BWARD contains a loop that enables multiple consultations to be done within one session. The items that need to be initialized once across several consultations are initialized in procedure START. Procedure LOOPSTART does the initialization that is needed before each consultation. In LOOPSTART, the rule group (set of rules constituting the knowledge-base) is queried, and the GOAL variables for this rule group are placed on a list. Next, a list of ANTECEDENT rules is obtained by looking for all rules with the ANTECEDENT property, and a list of *unconditional* ANTECEDENT rules is obtained. (Unconditional antecedent rules are rules in which the LHS does not

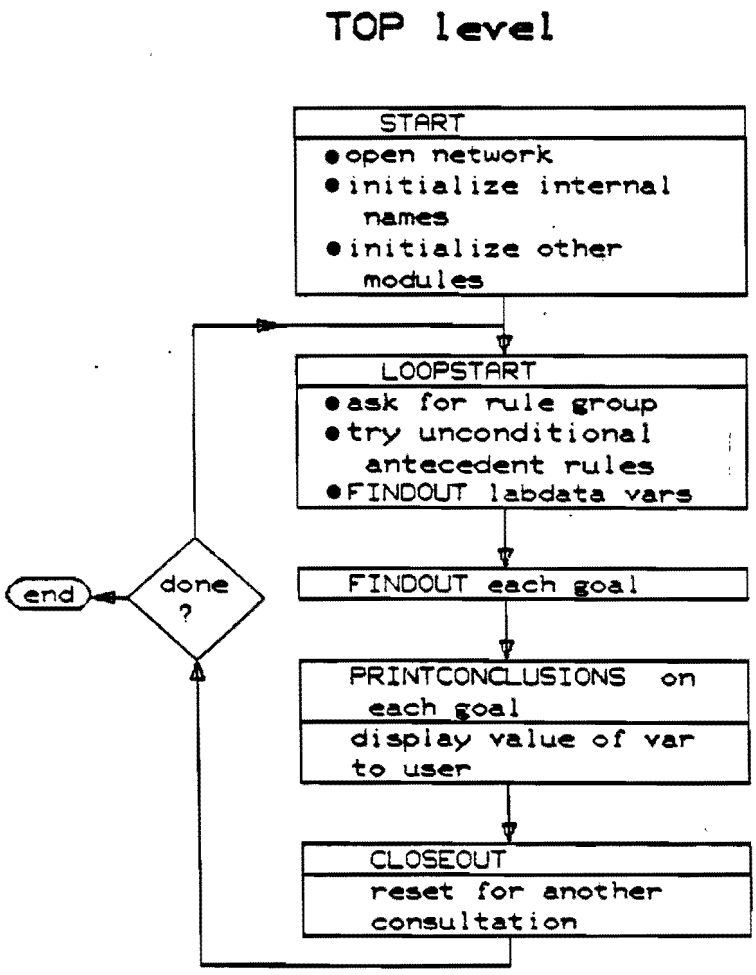


Figure 8. The flow chart for the top level of BWARD. The next two figures contain the flow charts for two important procedures of BWARD: FINDOUT and INFER.

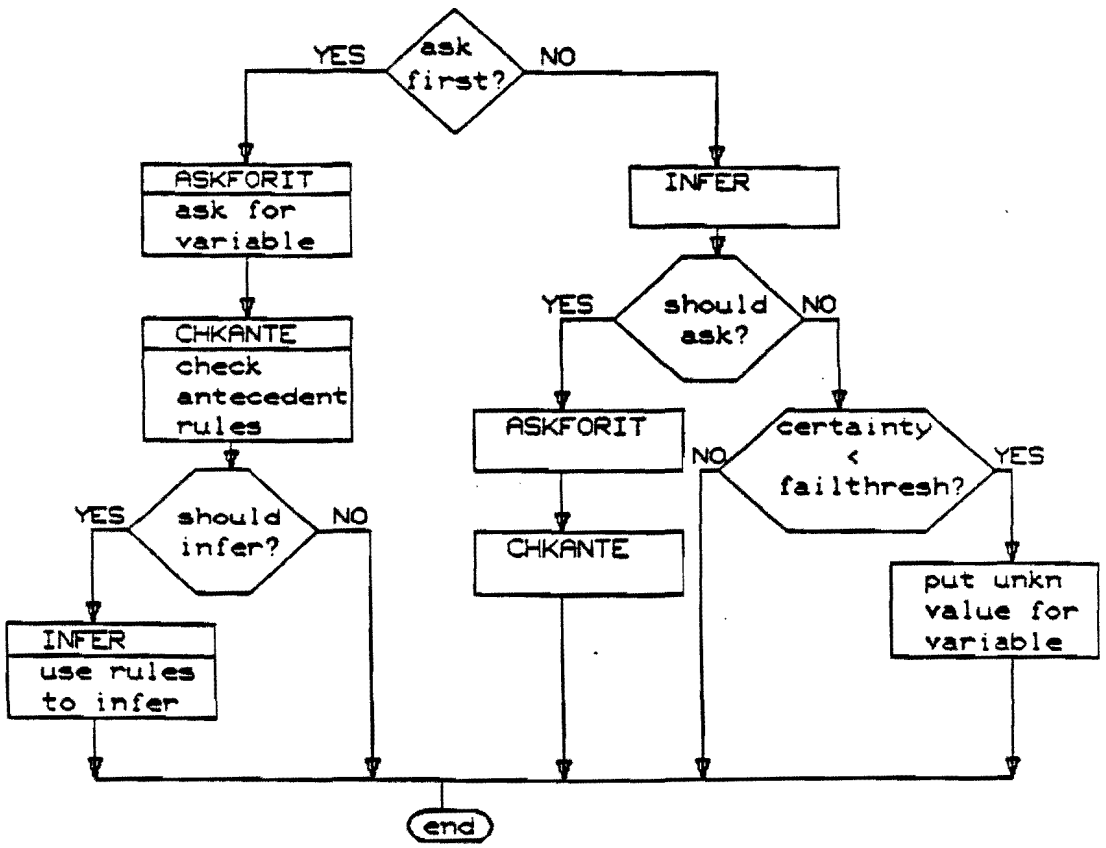


Figure 9. The flow chart for the FINDOUT procedure.

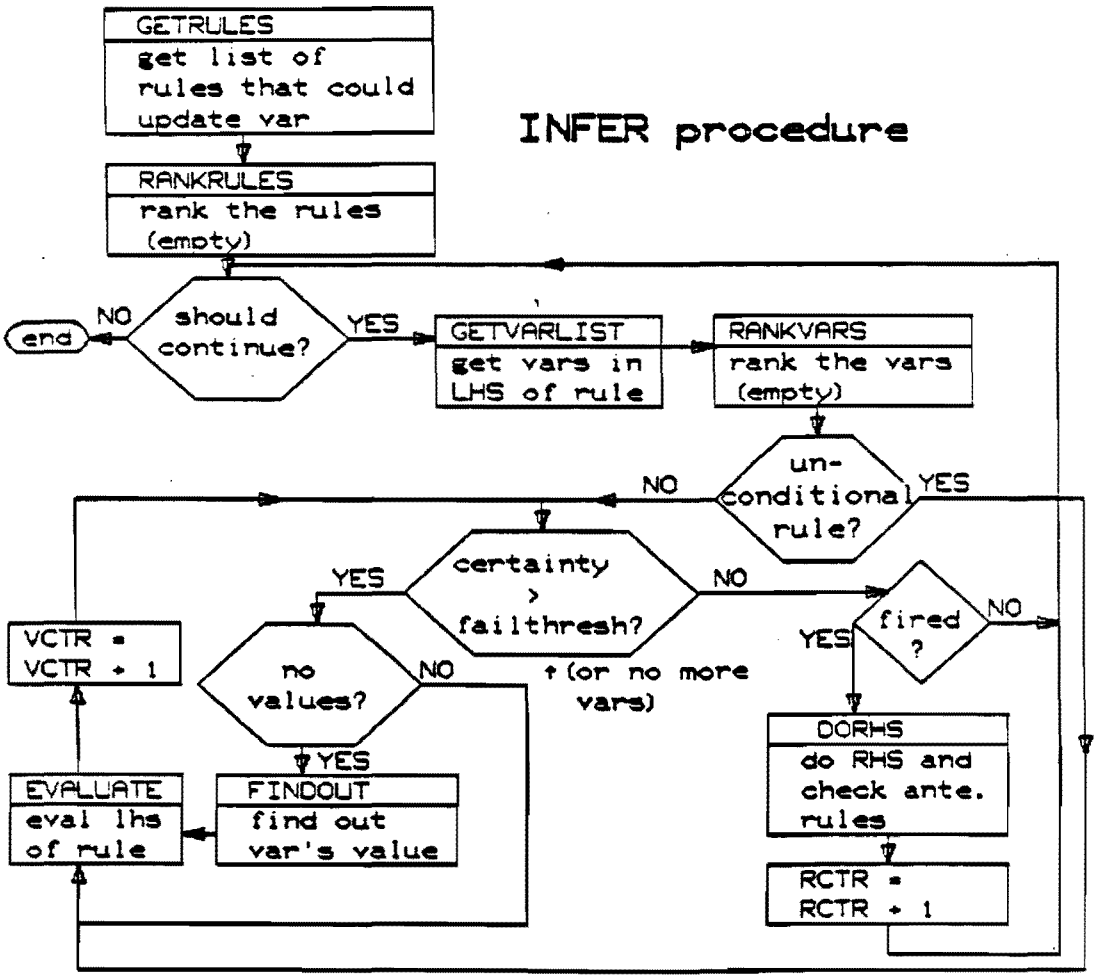


Figure 10. The flow chart for the INFER procedure.

depend on any variables and will be the first rules to fire.) A list of LABDATA variables is obtained next from the knowledge base. The unconditional ANTECEDENT rules are fired next. Finally, values for each of the LABDATA variables are obtained using the FINDOUT procedure described below.

After this initialization, each of the GOAL variables are found out using the FINDOUT procedure. This is the main part of the consultation. After the GOAL variables' values are obtained, they are printed with their confidences and the TRANS property of the GOAL variables using procedure PRINTCONCLUSIONS. The consultation is ended by cleaning-up for a new consultation and calling LOOPSTART again.

The FINDOUT procedure is responsible for finding the value of a variable; either by asking for it, or by using rules to infer it, or both. It first uses the ASKFIRST function to check if the current variable has the ASKFIRST property. If it does have the ASKFIRST property, then procedure ASKFORIT is called to solicit the value from the user. After obtaining the value from the user, procedure CHKANTE is called to evaluate and possibly fire any ANTECEDENT rules that have the variable in their LHS. The SHOULDINFER function is then called to see if the variable also needs to be inferred. The decision to infer a value depends on several factors. In the most general case there is the value/confidence pair obtained from the user as well as the value/confidence pair inferred from the rules.

The process of resolving any differences in the value/confidence pairs is called *voting*. There is a scalar Pascal variable, VOTESCHEME, that indicates the voting scheme. The voting method used in PLANT/cd is to replace the last value/confidence pair with the more current one. In the general case, the decision to infer after asking for the variable is a function of the voting scheme used. In PLANT/cd the variable is inferred after asking for it if the maximum confidence for the value(s) of a variable is below SATISFYTHRESHOLD. (It is possible to have several value/confidence pairs for the same variable.)

If a variable should not be asked first, then rules are used to possibly infer a value. After the attempt to get a satisfactory value by using rules, the SHOULDASK function is called to see if ASKFORIT needs to be called. (It should be realized that under some circumstances it might be useful to validate the results of using rules with the user. SHOULDASK provides a means of implementing such a strategy.) To be asked, the variable has to have a PROMPT property. Also the maximum certainty has

to be less than SATISFYTHRESHOLD. If the variable should be queried, then ASKFORIT as well as CHKANTE are called. If the variable should not be queried and the maximum certainty is below the FAILTHRESHOLD then UNKNOWN is placed as the value for the variable.

The INFER procedure is responsible for obtaining the value of a variable by using rules to infer it. INFER first gets the list of rules whose RHS update the current variable. This is done by procedure GETRULES. Next, procedure RANKRULES is called to order this list. This procedure is currently null. At this point a loop is set up to try the rules in the above list. SHOULDCONTINUE is called to check if the maximum of the confidences for the current variable's values is above SATISFYTHRESHOLD. This is one terminating condition for the loop; the other is running out of rules.

Inside the loop, a list of variables used in the LHS of the current rule is obtained by calling GETVARLIST. This list is ordered by RANKVARS. RANKVARS is currently null. There is a check after getting the list of variables for the current rule whether the rule is unconditional. If it is, then the LHS is evaluated[2] and the rule is checked to see if it fired. For rules that are not unconditional, then another loop is set up to use the FINDOUT procedure on each of the variables in the LHS variable list. For each variable that is found out, the LHS is reevaluated to see if the certainty of the LHS falls below FAILTHRESHOLD. If it does, the loop is terminated, and the rule will not fire. This terminating condition will only work for rules with conjunction only, and when the semantics for conjunction is the *min* function. The judgement of when to terminate the evaluation of a rule should be made by the rule evaluator, but in the current implementation, no such status is passed back. The other terminating condition for the loop is running out of variables.

Upon exiting the loop, the rule is checked to see if it fired, using the function FIRED. If the certainty of the LHS is greater than FIRETHRESHOLD, then DORHS is called to execute the RHS. Since executing the RHS may update variables, CHECKANTE is also called on all the variables that are updated in the RHS of the current rule. To see how this control scheme operates with an actual set of rules, see Section 5.1.

---

[2]It should be noted that even though the RHS is unconditional it may have side effects and therefore needs to be evaluated.

CHAPTER 5

IMPLEMENTATION OF PLANT/CD IN ADVISE

This chapter discusses the integrated PLANT/cd system. The PLANT/cd system consists of the backward chaining control scheme (BWARD), the rule base, and the special functions (TRAP) module (developed for PLANT/cd). If there are changes to the PLANT/cd rules, then the following procedure should be followed:

(1) Make any changes to the rules with a text editor and then use the ADVISE system RULE PARSER to parse the new rules.

(2) Add rule group information. This is done by adding two tuples to the knowledge base generated by the RULE PARSER. (The knowledge base is represented in the form of tuples.) These two tuples are in the form:

(RGRPS RGRPIDS RULEGROUP **GOALS).**

and

(GOALS GOALLIST GOAL1 GOAL2 GOAL3 ... GOALn**)**

where

RGRPS,RGRPIDS,GOALS, and **GOALLIST** are nodes whose printnames (RGRPS,RGRPIDS,GOALS and GOALLIST) are in the tuple manager dictionary so that they can be indexed by their printname,

RULEGROUP is the node representing the rule group for the PLANT/cd rules, and

GOAL1 GOAL2 GOAL3 .. GOALn are the nodes that are the goal variables of the rule group.

(3) If necessary, modify and/or add new functions to the TRAP module and recompile it.

(4) If necessary, make changes to the control scheme and recompile it.

Also the temperature data base needs to be updated on the VAX/780. The data comes from the temperature data base on the CYBER 175 in UN=3NHSNHS. A write-up on how this data is maintained is available in file TEXT1 on UN=3NHSNHS. A continuously maintained database for the current year's

data is on the file NWPRJ in the same directory. If the program is run on the VAX with data from the current year, an updated version of the file from the CYBER should be put on the VAX in directory *tempdata* under the *advise* directory.

5.1. The Plant/cd Rules to Predict Extent of Damage

The version of PLANT/cd that predicts amount of corn damage due to cutworms has been implemented in ADVISE. The rules for PLANT/cd were written with the extension of GVL_1 that is accepted by the RULE PARSER. As can be seen in Section 3.4, the major extension to GVL_1 used in PLANT/cd are special functions, called TRAP functions. A TRAP function is an escape mechanism to a Pascal case statement implemented as a separate ADVISE module, called the special functions or TRAP module. An example TRAP function call is:

[A=TRAP(1,B)]

where

[A=TRAP(1,B)] is a selector with a TRAP function in the reference place of the selector,

A is a variable that will be assigned the value of the TRAP function,

TRAP(1,B) is the TRAP function,

1 is the TRAP number to dispatch to in the Pascal case statement, and

B is an argument to be passed to the Pascal routine that implements the TRAP function.

When this selector is evaluated, the internal mark for the keyword TRAP is recognized by the RULE EVALUATOR, and it passes the TRAP number (the first argument) and the rest of the arguments to the TRAP module. The TRAP number serves as a case selector in the TRAP module. In PLANT/cd, TRAP functions are used for a linkage between the *deep* knowledge of the domain (the simulation routines) and the *surface* knowledge encoded in the rules. TRAP functions are also used for displaying tables and special user input. More details of the PLANT/cd TRAP module will be described in the next section.

Also used in PLANT/cd is the meta-value, *KNOWN,* for variables. This meta-value is used to to test for a value known with sufficient confidence and if not, cause the process of backward chaining for

the value. The meta-value, *INITIALIZED* is also used to make sure a variable is initialized. Currently these two values are not implemented, however the values *UNKNOWN* represented by the character "$" and *UNINITIALIZED* represented by the character "?" are implemented. These values are used in selectors with the *not equal* relational operator (<>) to get the same effects as using the meta-values *KNOWN* and *INITIALIZED* with the *equal* relational operator (=). Meta-values are used for selectors that operate at the control level.

The rule base operation will be described by "walking through" the *inference networks* (Figures 11 and 12). These inference networks illustrate the conceptual relationship between the rules and the variables in a rule base. In these figures the term *regular* variable means variables that are neither LABDATA nor GOAL variables. Table 1 contains the variables used in this rule base. In this table the variables' names, domains, and meanings are listed. Appendix A.2. contains the variable listing that can be parsed by the RULE PARSER. As described in the documentation of the backward chaining control scheme (Chapter 4), the first variables that are found out using the FINDOUT procedure are the LABDATA variables. These variables always need to be found out and are therefore processed first. They are not GOAL variables in that their values are never displayed to the user. For this knowledge base, there are four labdata variables, OBDATE, PLANTDATE, STACODE, and VARIETY. These variables are marked with the property LABDATA in the variable listing. The inference network associated with these variables is shown in Figure 11.

The value of OBDATE (observation date) is provided by invoking rule:

```
RULE1
TRUE ::> [OBDATE = TRAP(21,OBDATE,1,0)]; !GET OBDATE
```

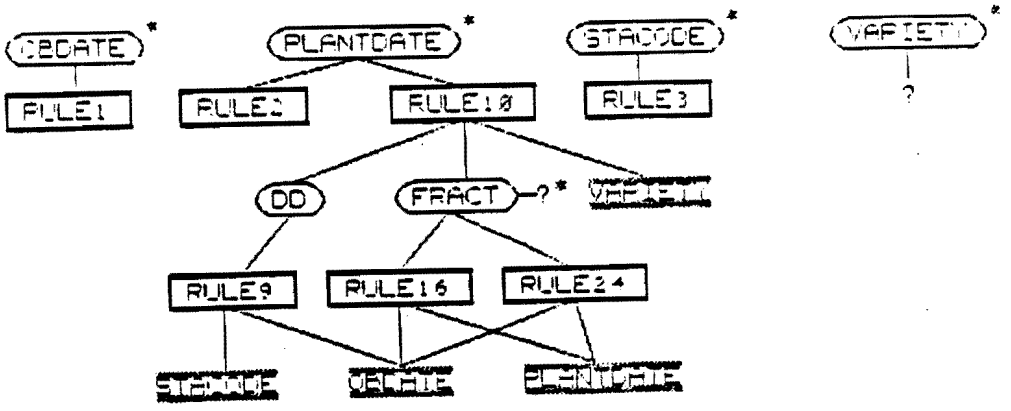
which has a *unconditional* (in the sense used in Chapter 4) LHS and therefore fires. The action that is done is to call TRAP function number 21 which handles getting dates from the user. The third argument to the TRAP function, 1, indicates that the year should be entered by the user. The fourth argument, 0, indicates that *UNKNOWN* is not accepted as an answer to OBDATE. OBDATE is an argument to the TRAP function so that the TRAP function can use OBDATE's *PROMPT* property to display to the user

Table 1. This table summarizes the variables used in PLANT/cd.

PLANT/cd Variables

Name	Domain	Meaning
		Asked Variables
OBDATE*	1..366	THE DATE FOR WHICH THE FIELD IS OBSERVED
PLANTDATE*	1..366	THE DATE FOR PLANTING CORN
STACODE*	1000..9999	THE STATION CODE
FRACT	0.0..10.0	THE OBSERVED FRACTIONAL LEAF STAGE OF THE CORN
VARIETY*	FULLSEASON,MIDSEASON	THE CORN VARIETY
MOISTURE	WET,DRY,NORMAL	THE SOIL MOISTURE IN THE FIELD
HOTWINDY	YES,NO	IT IS HOT AND WINDY
MUCHWEEDS	YES,NO	THERE ARE GREATER THAN 4 WEEDS/FOOT OF ROW
		Trap Function Output & Result Variables
YIELD1	0.0..100.0	THE YIELD W/O INSECTICIDE TREATMENT
YIELD2	0.0..100.0	THE YIELD WITH INSECTICIDE TREATMENT
YIELDINC1	0.0..100.0	THE INCREASE IN YIELD DUE TO RECOVERY W/O INSECTICIDE TREATMENT
YIELDINC2	0.0..100.0	THE INCREASE IN YIELD DUE TO RECOVERY WITH INSECTICIDE TREATMENT
TYIELD1	0.0..100.0	THE TOTAL YIELD W/O INSECTICIDE TREATMENT
TYIELD2	0.0..100.0	THE TOTAL YIELD WITH INSECTICIDE TREATMENT
CORNDAMAGE	0.0..100.0	PERCENT TOTAL DAMAGE OF THE CORN
TOTRECOVER	0.0..100.0	PERCENT TOTAL RECOVERY OF THE CORN
STAND	0.0..100.0	PERCENT FINAL STAND OF THE CORN
MTRAP	YES,NO	MOTH TRAP COUNTS HAVE BEEN GOTTEN
EGGS	YES,NO	EGGS HAVE BEEN COMPUTED
FEGGS	YES,NO	EGGS IN THE FIELD HAVE BEEN COMPUTED
ATTR	YES,NO	ATTRACTIVENESS RATINGS OF THE FIELD HAVE BEEN GOTTEN
BCWPOP	YES,NO	BLACK CUTWORM LARVAL COUNTS HAVE BEEN GOTTEN
DD	YES,NO	DEGREE-DAY TABLE HAS BEEN READ IN
CD	YES,NO	CORN DEVELOPMENT HAS BEEN SIMULATED
BCWD	YES,NO	BLACK CUTWORM DEVELOPMENT HAS BEEN SIMULATED
POPVSSTAGE	YES,NO	THE BLACK CUTWORM VS LEAFSTAGE TABLE HAS BEEN COMPUTED

* denotes LABDATA variables



KEY

⬭ - regular variable	? - value of the variable could be asked
⬭* - labdata variable	?* - value of the variable will be asked
═ - variable mentioned elsewhere in network	▭ - backward-executed rule
⬭** - goal variable	▭* - forward-executed rule (antecedent rule)
	→ - side effect

Figure 11. The inference network for the LABDATA variables in PLANT/cd. The key for the networks is below.

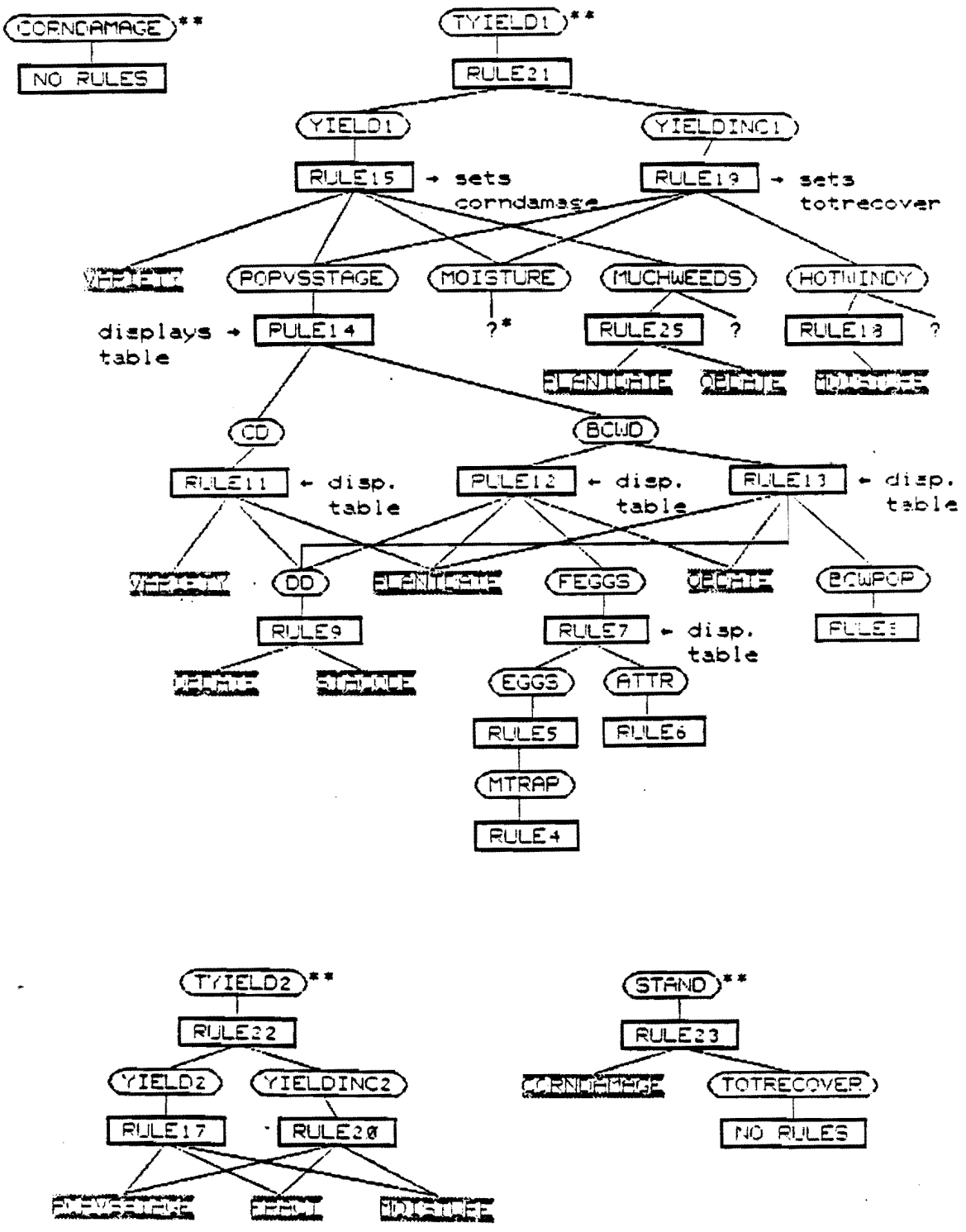


Figure 12. The inference network for the GOAL variables in PLANT/cd. The goal variables in order are: *CORNDAMAGE, TYIELD1, TYIELD2, and STAND.*

when soliciting OBDATE's value. (Each of the TRAP functions and their parameters are explained in the next section.)

The value for PLANTDATE (planting date of the corn) can be found by RULE2 or RULE10. The rule:

```
RULE2
TRUE ::> [PLANTDATE = TRAP(21,PLANTDATE,0,1)]; !GET PLANTDATE
```

which has a unconditional LHS is tried first and the RHS calls TRAP function number 21. The third argument, 0, indicates that the year is not needed for this date. The fourth argument, 1, indicates that, for this date request, the user can provide the value UNKNOWN. In such a case rule:

```
RULE10
[DD = KNOWN][OBDATE = KNOWN][FRACT = KNOWN][VARIETY = KNOWN] ::>
  [PLANTDATE = TRAP(13,OBDATE,FRACT,VARIETY,DD)];
!ESTIMATE PLANTING DATE FROM FRACTIONAL LEAF STAGE
```

is used to estimate the planting date from the degree-day (DD) table and the observed fractional leaf stage of the corn (FRACT). This is done by executing TRAP 13 which simulates plant development backwards, given the DD table, FRACT, and VARIETY (the corn variety). The variable DD is found out (i.e., inferred) from the observation date (OBDATE) and the identifier of the closest weather station (STACODE).

```
RULE9
[OBDATE = KNOWN][STACODE = KNOWN] ::>
  [DD = TRAP(12,OBDATE,STACODE)];
!GET DEGREE-DAY TABLE.
```

RULE9 invokes TRAP 12 to get the degree-day data. The variable STACODE (the identifier of the closest weather station) is provided by using RULE3. RULE3 uses TRAP 2 to get STACODE:

RULE3
TRUE ::> [STACODE = TRAP(2)]; ! GET STATIONCODE.

TRAP 2 is a special procedure to ask for weather stations.

Continuing to analyze RULE10, FRACT is queried since the rules will provide no value in this context even though it has rules to infer its value. (If FRACT is needed later these rules can be used. These rules will be explained later.) Finally, VARIETY is queried.

If STACODE is not provided a value while finding PLANTDATE, then it is the next LABDATA variable to be found out. As explained before, it is supplied a value by RULE3. Finally, the last LABDATA variable is VARIETY, and like STACODE, its value could have been provided while inferring the value for PLANTDATE.

The next variables to be found out are the goal variables, CORNDAMAGE, TYIELD1, TYIELD2, and STAND. The inference network for these is shown in Figure 12. The value of these variables is the goal of the consultation and consequently their values are printed out as the results of the consultation at its end. These variables are put in a list of goals for the BCW damage rule group in the knowledge base.

The first goal, CORNDAMAGE (percent total corn damage), has no rules that update it directly. Rather, it is provided a value by side effect when RULE15 fires (see below).

The second goal, TYIELD1 (corn yield without insecticide treatment), is computed by adding to YIELD1 (corn yield without regrowth considered) to YIELDINC1 (the increment to the yield because of regrowth). This computation is done by rule:

RULE21
[YIELD1 = KNOWN][YIELDINC1 = KNOWN] ::>
[TYIELD1 = YIELD1+YIELDINC1];
!COMPUTE TOTAL YIELD W/O INSECTICIDE TREATMENT.

RULE21 causes YIELD1 and YIELDINC1 to be found out. YIELD1 is computed by invoking TRAP 17 in rule:

```
RULE15
[POPVSSTAGE = KNOWN][MUCHWEEDS = KNOWN]
[VARIETY = KNOWN][MOISTURE = KNOWN]        ::>
  [YIELD1 = TRAP(17,MUCHWEEDS,VARIETY,MOISTURE,
                 0.25,CORNDAMAGE,POPVSSTAGE)];
!ESTIMATE YIELD W/O INSECTICIDE TREATMENT
```

TRAP 17 computes YIELD1 from the variables MUCHWEEDS,VARIETY,MOISTURE, and POPVSSTAGE (whether the Black Cutworm vs. leafstage table has been computed). The constant 0.25 specifies the parasitism rate used in the yield computation. (The parasitism rate is the mortality rate of Black Cutworm larvae due to parasites). The variable CORNDAMAGE gets set when TRAP 17 is called. This is the side effect mentioned earlier. The variable MUCHWEEDS (whether the weeds are greater than 4 weeds/foot of row) is supplied a value by rule:

```
RULE25
[OBDATE <= PLANTDATE] ::>
  [MUCHWEEDS = NO];
!ASSUME NO WEEDS IF PROJECTING BEFORE PLANTDATE
```

or by asking for it. RULE25 is fired if the observation date (OBDATE) is less than or equal to the planting date (PLANTDATE). In this case MUCHWEEDS is assumed to be false. If the observation date is less than or equal to the planting date, then BCW damage is being projected before corn planting has occurred and assuming the value NO for MUCHWEEDS is reasonable. If RULE25 does not fire, then MUCHWEEDS is queried.

Returning to RULE15, the variable MOISTURE (soil moisture in the field) is queried. The value of POPVSSTAGE is computed by the inference net starting from RULE14. This inference net invokes the core of the simulation procedures. The value of YES for POPVSSTAGE represents the fact that the population vs. stage matrix has been computed. This is the key matrix generated by the simulation procedures. This matrix represents snapshots of the larval age spectrum at different stages of corn development (leaf stage emergence). Once cutworm development is placed in terms of corn development, then TRAP 17 can estimate the damage by multiplying the population vs. leafstage matrix by a

"munchability" matrix which indicates the number of different-aged corn plants different-aged larvae can eat. This is approximately how TRAP 17 works. The population vs. leafstage matrix is matrix K in [Troester et al. 1982b].

A value for POPVSSTAGE is provide by rule:

```
RULE14
[CD = KNOWN][BCWD = KNOWN] ::>
  [POPVSSTAGE = TRAP(16,CD,BCWD)]TRAP(11);
!COMPUTE POP VS STAGE MATRIX AND DISPLAY[1]
```

RULE14 invokes TRAP 16 which implements the process of putting instar transitions in terms of corn leaf stage emergence. This is the synchrony process in Figure 6. RULE14 also invokes TRAP 11 which displays the population vs. leafstage matrix to the user. The values of CD (corn development) and BCWD (Black Cutworm development) are needed for RULE14 to fire. The value for CD is provided by rule:

```
RULE11
[DD = KNOWN][PLANTDATE = KNOWN][VARIETY = KNOWN] ::>
  [CD = TRAP(14,PLANTDATE,VARIETY,DD)]TRAP(10);
!RUN SIMULATION PROGRAM FOR CORN DEVELOPMENT AND DISPLAY RESULTS
```

RULE11 invokes TRAP 14 which simulates corn development. TRAP 10 is also executed to display the results of corn development. RULE11 requires values for VARIETY, DD, and PLANTDATE.

BCWD is provided a value by RULE12 or RULE13. RULE12 is fired if a prediction is being made before planting date, and RULE13 is fired if the prediction is after planting date. In rule:

```
RULE12
[DD = KNOWN][OBDATE < PLANTDATE][FEGGS = KNOWN] ::>
[BCWD = TRAP(15,OBDATE,PLANTDATE,FEGGS,DD)]TRAP(9,OBDATE);
!RUN SIMULATION PROGRAM FOR BCW DEVELOPMENT AND DISPLAY RESULTS
```

[1] In this rule TRAP 11 functions as a selector and is used for its side effect to display the population vs. leafstage matrix.

TRAP 15 is invoked to simulate Black Cutworm development given FEGGS (the history of the eggs laid in the field), OBDATE, PLANTDATE, and DD. TRAP 9 is used to display the results of cutworm development. FEGGS is provided a value by rule:

```
RULE7
[EGGS = KNOWN][ATTR = KNOWN] ::> [FEGGS = TRAP(7,EGGS,ATTR)]TRAP(8);
!CONVERT TO EGGS IN FIELD AND DISPLAY
```

RULE7 calls TRAP 7 to compute the eggs laid in the field given ATTR (a history of the field attractiveness rating) and EGGS (a history of the eggs laid in the region). It also calls TRAP 8 to summarize for the user the number of eggs laid in the region and in the field. ATTR is supplied by rule:

```
RULE6
TRUE ::> [ATTR = TRAP(4)]; !GET ATTRACTIVENESS RATINGS
```

which is a unconditional rule and calls TRAP 4 to get the attractiveness rating vector. EGGS is updated by rule:

```
RULE5
[MTRAP = KNOWN] ::> [EGGS = TRAP(6,MTRAP)]; !CONVERT TO EGGS
```

which invokes TRAP 6. TRAP 6 converts MTRAP (moth trap counts) to EGGS. MTRAP is updated by rule:

```
RULE4
TRUE ::> [MTRAP = TRAP(3)]; !GET MOTH TRAP COUNTS
```

which is a unconditional rule and calls TRAP 3 to get the moth trap counts.

The rule:

```
RULE13
[DD = KNOWN][OBDATE >= PLANTDATE][BCWPOP = KNOWN] ::>
  [BCWD = TRAP(15,OBDATE,PLANTDATE,BCWPOP,DD)]TRAP(9,OBDATE);
!RUN SIMULATION PROGRAM FOR BCW DEVELOPMENT AND DISPLAY RESULTS
```

is fired when the prediction is after planting date. It calls TRAP 15, as in RULE12, but it uses BCWPOP (Black Cutworm population) instead of FEGGS. (See next section for more details of this TRAP function.) RULE13 also calls TRAP 9 for display of cutworm simulation results. The value of BCWPOP is provided by rule:

```
RULE8
TRUE ::> [BCWPOP = TRAP(5)]; ! GET LARVAL COUNTS
```

which is a unconditional rule and invokes TRAP 5 to get the larval counts.

Returning to analyze RULE21, YIELDINC1 is provided a value by rule:

```
RULE19
[POPVSSTAGE = KNOWN][MOISTURE = KNOWN][HOTWINDY = KNOWN] ::>
  [YIELDINC1 = TRAP(19,MOISTURE,HOTWINDY,TOTRECOVER,POPVSSTAGE)];
!COMPUTE YIELD INCREASE W/O INSECTICIDE TREATMENT.
```

RULE19 invokes TRAP 19 which computes the yield increment without insecticide treatment due to recovery. TRAP 19 uses POPVSSTAGE, MOISTURE, MUCHWEEDS, and HOTWINDY. HOTWINDY can be inferred or queried. The rule:

```
RULE18
[MOISTURE = DRY] ::> [HOTWINDY = IRRELEVANT];
!IF MOISTURE IS DRY, THEN HOTWINDY DOESN'T MATTER
```

sets the value of HOTWINDY to IRRELEVANT if MOISTURE is DRY.[2] If this is not the case, then HOTWINDY is queried.

---

[2]This is done so that HOTWINDY is not queried when MOISTURE is DRY, since HOTWINDY's value becomes IRRELEVANT. This meta-value has not been implemented yet and currently the value NO is used.

The third goal, TYIELD2 (yield with insecticide treatment), is treated much like TYIELD1 in the inference network that updates it. The rule:

```
RULE22
[YIELD2 = KNOWN][YIELDINC2 = KNOWN] :>
  [TYIELD2 = YIELD2+YIELDINC2];
!COMPUTE TOTAL YIELD WITH INSECTICIDE TREATMENT
```

provides TYIELD2 with a value. RULE22 requires YIELD2 (yield without recovery with insecticide treatment) and YIELDINC2 (yield increment due to recovery with insecticide treatment). YIELD2 is updated by rule:

```
RULE17
[POPVSSTAGE = KNOWN][FRACT = KNOWN][MOISTURE = KNOWN] ::>
  [YIELD2 = TRAP(18,FRACT,MOISTURE,POPVSSTAGE)];
!COMPUTE YIELD WITH INSECTICIDE TREATMENT
```

which calls TRAP 18 to do the calculation. YIELDINC2 is updated by rule:

```
RULE20
[POPVSSTAGE = KNOWN][FRACT = KNOWN][MOISTURE = KNOWN] ::>
  [YIELDINC2 = TRAP(20,FRACT,MOISTURE,POPVSSTAGE)];
!COMPUTE YIELD INCREASE WITH INSECTICIDE TREATMENT
```

which calls TRAP 20 to do the calculation. TRAP 18 and TRAP 20 require FRACT, MOISTURE and POPVSSTAGE. If FRACT was not provided a value earlier, it will be provided a value by RULE16 or RULE24. The rule:

```
RULE24
[OBDATE > PLANTDATE] ::>
 [FRACT = TRAP(22,PLANTDATE,OBDATE)];
! COMPUTE FRACTIONAL LEAF STAGE
```

provides FRACT (fractional corn leaf stage) a value if cutworm damage is being projected from larval counts (hence OBDATE > PLANTDATE). This is done by TRAP 22. If RULE24 is not used, FRACT

will be updated by rule:

```
RULE16
[PLANTDATE = INITIALIZED][OBDATE <= PLANTDATE] ::> [FRACT = 0.0];
!LEAF STAGE IS ZERO FOR PREDICTING FROM MOTH CASE
```

which sets FRACT to zero if a prediction is being made before planting date. (The first selector prevents this rule from firing if FRACT is sought in the process of providing a value to the LABDATA variable PLANTDATE.)

The fourth goal, STAND (final plant stand), is provided a value by rule:

```
RULE23
[CORNDAMAGE = KNOWN][TOTRECOVER = KNOWN] ::>
  [STAND = 100.0 - (CORNDAMAGE-TOTRECOVER)];
!COMPUTE STAND
```

RULE23 requires CORNDAMAGE (the first goal) and TOTRECOVER (percent total recovery of the corn). TOTRECOVER is updated as a side effect when RULE19 fires.

Appendix A contains rule and variable listings for PLANT/cd. The format of the rule and variable definitions is what is currently accepted by the ADVISE system RULE PARSER. In those listings the characters, "$" for *UNKNOWN* and "?" for *UNINITIALIZED* are used.

5.2. The Special Functions (TRAP) Module for PLANT/cd

This module contains 21 TRAP functions used in PLANT/cd. These functions implement the *deep* model of BCW damage as developed by Steve Troester [Troester et al. 1982a,b,c] [Troester et al. 1983]. The BCW development model by Kimpel [Kimpel 1978] which was used in Steve Troester's programs [Troester et al. 1982c], [Troester et al. 1983], was not used in PLANT/cd. Instead, a simplified simulation was developed. This is explained in the description of TRAP 15.

5.2.1. Architecture of the Special Functions (TRAP) Module

As seen in the technical level diagram of ADVISE in Figure 2, the RULE EVALUATOR uses the special functions module (also known as the TRAP module) to evaluate TRAP functions that are used in the rules. A TRAP function has the form:

TRAP(number,p1,p2,...,pn)

where

number is the TRAP number, and

p1,p2,...,pn are the parameters of the TRAP function.

The RULE EVALUATOR passes the TRAP number and the parameters to procedure TRAPFUNC in the TRAP module. Procedure TRAPFUNC contains a case statement in which the TRAP numbers serve as case labels. Once a case label is branched to, several things are done. The incoming parameters are converted from their tuple representation to a Pascal representation. Parameters that are nominal are usually mapped into a Pascal scalar variable. Next the statements that make up the definition of the function are executed. This usually consists of a call to a Pascal procedure or function. Finally the outgoing parameters are converted form their Pascal representation to their tuple representation. These procedures are scoped within procedure TRAPFUNC although there are several utility procedures outside. One procedure that is outside is procedure TRINIT which the control scheme (BWARD) calls to initialize the TRAP module. Below is a description of the TRAP functions:

- *TRAP 1.* This function was used for developing the TRAP module and is no longer used.
- *TRAP 2.* This function calls TRASKSTATION which gets the name of the closest weather station from the user by a special protocol. This procedure is necessary because STACODE (the station code) is a nominal type variable with too many domain elements to be handled by the normal method of creating a set of touch sensitive areas on the screen.
- *TRAP 3.* This function calls TRASKTRAP to obtain the moth trap counts over a span of 11 weeks beginning March 17. The user can enter *unknown* for any of the entries and the system will assume a

value of 0. Giving an unknown value will also set the certainty of the first parameter passed to the TRAP function to 0.5. Also provided is a check for proper number format. If something entered is not a number, the the user is asked to renter his data. This check is provided on all the user input routines in the TRAP module.

- *TRAP 4.* This function calls TRASKATTR to obtain the field attractiveness rating for the same 11 week time span. Unknown values are handled in the same way as in TRASKTRAP.
- *TRAP 5.* This function calls TRASKBCWPOP to get the larval age spectrum in the field. This works like TRASKTRAP and TRASKATTR. TRASKTRAP, TRASKATTR, and TRASKBCWPOP use TRDSPWEEKS and TRDSPLCNTS to produce the appropriate column headings. They also use TRGSTUFF to get the user input.
- *TRAP 6.* This function calls TRCTOEGGS to convert the moth trap counts to an egg population. This is done by multiplying the moth trap count by 5.7. This number was derived by Steve Troester.
- *TRAP 7.* This function calls TRCTOFLD to convert the eggs laid in the region to eggs laid in the consultation field. This is done by multiplying the egg population history by the attractiveness rating history.
- *TRAP 8.* This function calls TRDSPEGGS to display the egg population in the consultation field and the surrounding area.
- *TRAP 9.* This function calls TRDSPTAB to display the results of the cutworm development model. This is in the form of a transition matrix which shows the dates when different instars will undergo molts.
- *TRAP 10.* This function calls TRDSPLTAB to display the results of plant development.
- *TRAP 11.* This function calls TRDSPPOPVSSTAGE to display the results of putting cutworm development in terms of corn development. This creates the matrix K in [Troester et al. 1982b].
- *TRAP 12.* This function calls TRGETDDTABLE to get the degree-day data necessary for the corn and cutworm development models. The two incoming arguments, STACODE and OBDATE, are

used to specify the location and start date of the data, respectively.

- *TRAP 13.* This function calls TRPLANTDATE to trace backward the development of the corn to estimate the planting date (PLANTDATE) given the observed fractional leaf stage (FRACT), the corn variety (VARIETY), and the observation date (OBDATE). This computation relies on results described in [Troester et al. 1982a,c]. The procedure TRLEAFLOOKUP is used to access leaf emergence data.

- *TRAP 14.* This function calls TRCORNDEVELOP which is the corn development model. Inputs are the planting date (PLANTDATE) and the corn variety (VARIETY). This also relies on the work referenced above.

- *TRAP 15.* This function calls TRBCWDEVELOP which is the cutworm development model. Inputs are OBDATE and PLANTDATE. This model actually has two different parts. One section is only used if eggs are forward developed from an observation date less than planting date. This section is bypassed if actual larval counts are input. In Steve Troester's work, eggs are developed with a model built by Kimpel [Kimpel 1978] and was written in the simulation language GASP 5. For PLANT/cd another approach was taken and was approved by Steve Troester. In this approach the egg population is forward developed (using Troester's development rates) until planting date. Then the different-aged larvae are classified into the closest instar category. The simulation from there is the same as described in [Troester et al. 1982a,c] and is the same one that simulates cutworm development given actual larval counts. This procedure calls TRAVERAGE as a utility procedure. The procedures TRSMOOTH and TRSURVIVE are provided for experimenting with smoothing the egg distribution, and simulating larval mortality. Currently these are not used.

- *TRAP 16.* This function calls TRPOPVSSTAGE to put the results of cutworm development in terms of corn development. This is based on the work described in [Troester et al. 1982a,c].

- *TRAP 17.* This function calls TR1DAMAGE to estimate cutworm damage without recovery and without regrowth. Inputs to this function are MUCHWEEDS, VARIETY, MOISTURE, and the parasitism rate which is set to a constant. The output of this function consists of the corn yield

without insecticide treatment and without recovery (YIELD1) and the percent damaged corn (CORNDAMAGE). This function and the following three TRAP functions rely on work described in [Troester et al. 1982b].

- *TRAP 18.* This function calls TR2DAMAGE to estimate cutworm damage without recovery but with insecticide treatment. This function uses FRACT and MOISTURE (soil moisture) to compute YIELD2 (yield without recovery and with insecticide treatment).
- *TRAP 19.* This function calls TR1RECOVER to estimate corn recovery without insecticide treatment. This function uses MOISTURE and HOTWINDY (whether it is hot and windy) to compute YIELDINC1 (yield increase due to recovery without insecticide treatment) and TOTRECOVER (percent total recovery of the corn).
- *TRAP 20.* This function calls TR2RECOVER to estimate corn recovery with insecticide treatment. It uses FRACT and MOISTURE to compute YIELDINC2 (yield increase due to recovery with insecticide treatment).
- *TRAP 21.* This function is used to prompt for and obtain the value for the date-valued variables used in PLANT/cd. It returns the number of the day entered in *mm/dd/yy* format. To obtain the date from the user, it calls TRASKDATE. TRASKDATE checks for proper date format. The year is stored internally in the TRAP module.
- *TRAP 22.* This function calls TRGETLSTAGE which is used to calculate the fractional leaf stage (FRACT) given PLANTDATE and OBDATE.
- *TRAP testnum.* This function is used to call the TRAP module exerciser routine, TRTESTR. This was used by the author in the development of the TRAP module for PLANT/cd and was designed to aid him in debugging the TRAP module.

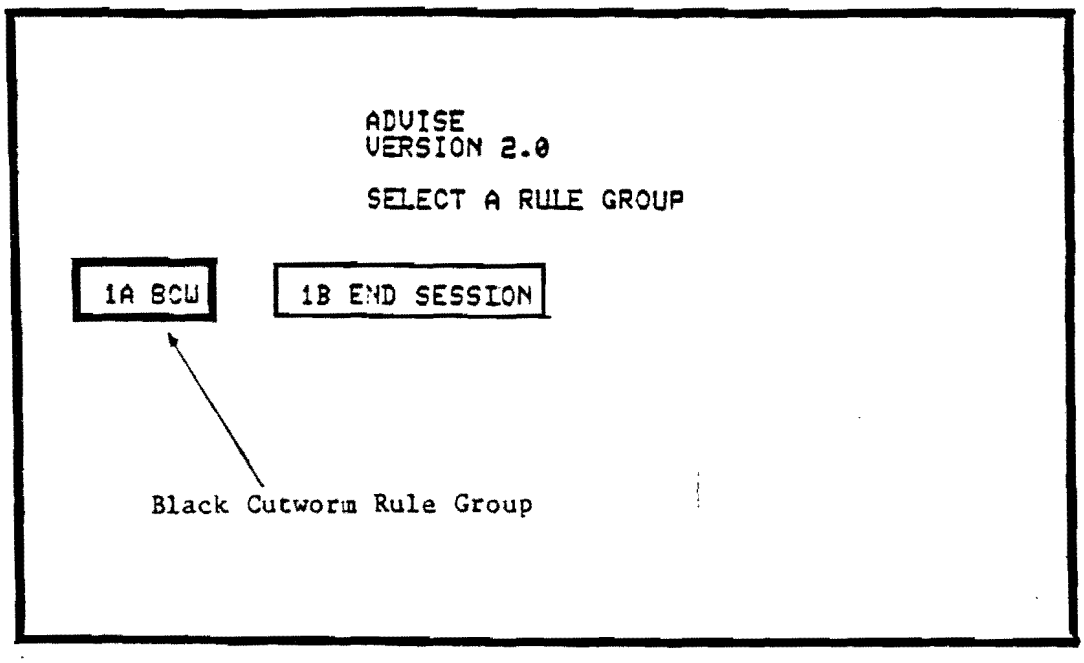
The TRAP module also has a main program that is used to run it in test mode. In this mode, the major functions are exercised, and damage estimates can be obtained without the use of the knowledge base (i.e., it runs in stand alone mode.) The main program calls TREXERCISE to do the exercising functions. This procedure calls TRAPFUNC and has it branch to *testnum* in the case statement.

5.3. A Session with Plant/cd

Figure 13 contains panels of a typical session with PLANT/cd. These panels were produced with a Gould electrostatic plotter attached to a Tektronix 4014 graphics terminal. A version of the DISPLAY MODULE is available for this terminal. It uses the cursor instead of a touch panel. These panels therefore depict closely what one would see on the ORION plasma panel display terminal. The rectangular areas within a panel are *touch targets* which the user can touch to provide a choice. The computer acknowledges the choice by making the border of the touched area larger. Touch targets can also be seen in Figure 3. A touch target in this state appears in the top panel in Page 63. Some questions require the user to type, as in the bottom panel in Page 61. This occurs when the protocol for generating touch targets and getting touched responses would be too cumbersome. This occurs in several cases:

- The variable that is being queried is an *interval* type variable. (Interval variables are specified to have a range of values, possibly large or infinite.)
- The domain elements of the variable being queried require a special format. This is the case for the date in the bottom panel of Page 61.
- The variable is a vector. This is illustrated in Page 64.

Sometimes output is displayed and the user needs time to read it. This is the case in the top panel of Page 66. The user is prompted to touch the screen when he/she is ready to continue.



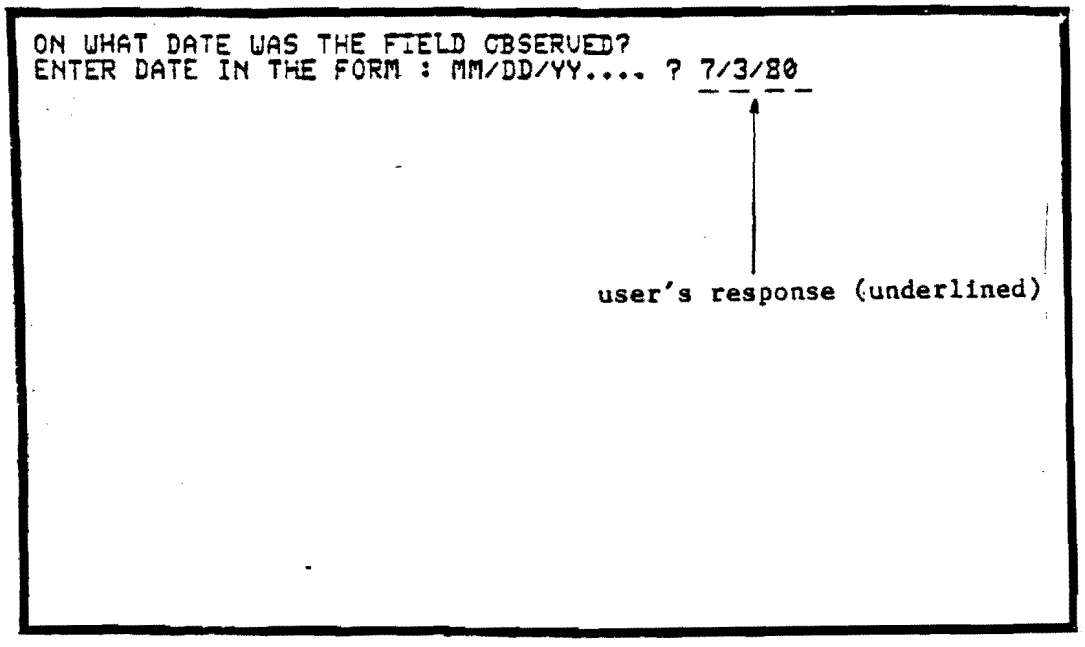


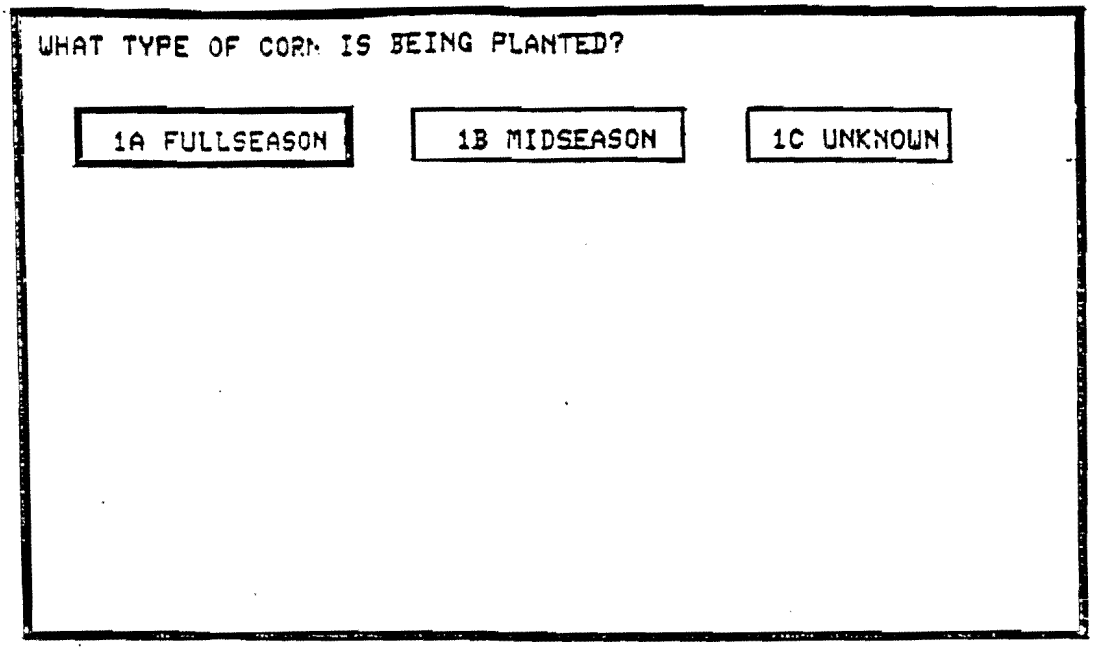
Figure 13. This figure, which is continued on the next seven pages, depicts a session with PLANT/cd. The top panel is the introductory panel for PLANT/cd in which the user is queried for the rule group. There is presently one, the Cutworm damage rule group. The user responds by touching one of the outlined regions. The user is being queried for the observation date in the next panel.

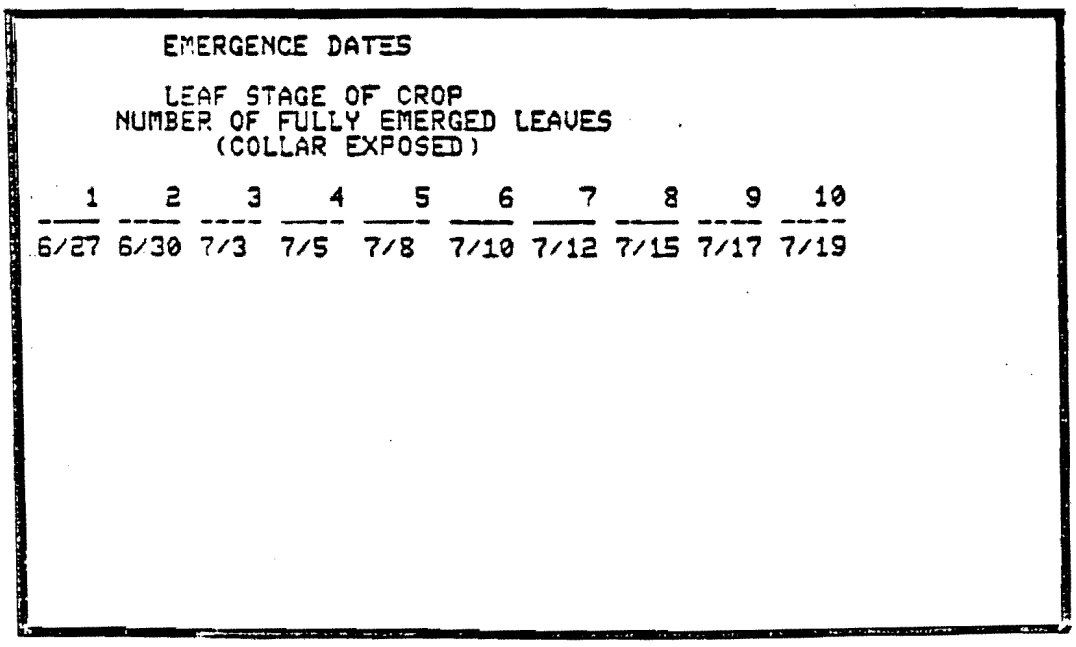
```
WHAT IS THE PLANTING DATE?
ENTER DATE IN THE FORM : MM/DD ....
OR PRESS CR IF YOU DON'T KNOW.. ? 6/28
```

```
 ENTER CLOSEST WEATHER STATION NAME
TYPE 'HELP' TO SEE LIST OF ALLOWED STATIONS
ENTER NAME... ? HELP

BROWNSTOWN      C GIRARDEAU     CARBONDALE
CHICAGO         COBDEN          DECATUR
DIXON SPRINGS   DUBUQUE         ELIZABETH
ELWOOD          HUMBOLDT        ILKA
KILBOURNE       MOLINE          PADUCAH
PEORIA          PERRY           QUINCY
ROCKFORD        ST LOUIS        SPRINGFIELD
TERRE HAUTE     TISKILWA        URBANA
VINCENNES       WATSEKA
ENTER NAME... ? IUKA
```

Figure 13 continued. In the top panel, the user is being prompted for the planting date. If the user does not know this date, the planting date will be inferred from the fractional leaf stage. The user is being prompted for the weather station in the bottom panel.





TOUCH SCREEN TO CONTINUE.

Figure 13 continued. The user is being asked for the corn variety in the top panel. The region that has a thicker outline indicates the user's response. The next panel displays the stages of corn development predicted by the system on the basis of planting date, corn variety, and temperature data for the region the field is in.

```
ENTER THE NUMBER OF BLACK CUTWORM MOTHS CAUGHT
PER TRAP DURING THE FOLLOWING WEEKS

    ENTER ONE VALUE UNDER EACH WEEK
    TO TERMINATE ENTERING PRESS THE CR KEY
    ENTER THE CR KEY ALONE WHEN NO MOTHS FOR
    THE WEEK ARE CAPTURED.....

WEEK BEGINNING:
3-17  3-24  3-31  4-7   4-14  4-21  4-28  5-5   5-12  5-19  5-26
----  ----  ----  ----  ----  ----  ----  ----  ----  ----  ----
0     0     0     6.5   _     10    5     _     -     -     -
```

user's responses

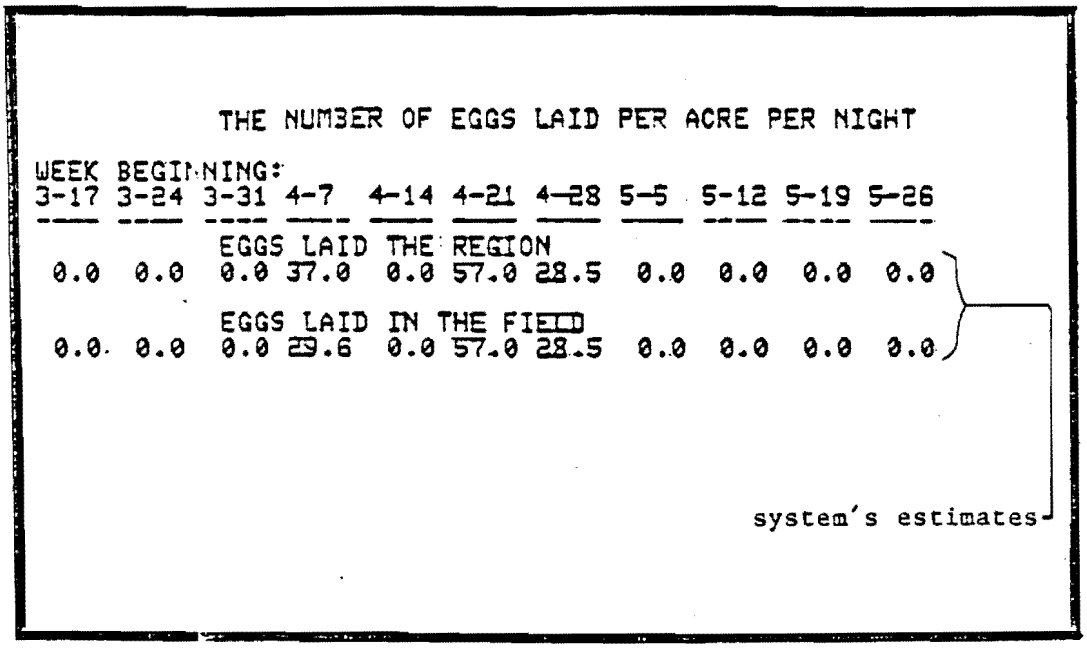
```
ENTER THE FIELD ATTRACTIVENESS RATING FOR EGG
LAYING DURING THE FOLLOWING WEEKS.

   ENTER A VALUE BETWEEN 0.0 AND 1.0 UNDER EACH
   WEEK.

WEEK BEGINNING:
3-17  3-24  3-31  4-7   4-14  4-21  4-28  5-5   5-12  5-19  5-26
----  ----  ----  ----  ----  ----  ----  ----  ----  ----  ----
0     0     .4    0.8   1     1     1     1     1     1     1
```

user's responses

Figure 13 continued. The user is prompted for the moth trap counts and attractiveness ratings in these panels. The underscores in the top panel indicates that the user did not have moth trap catches for some of the weeks.

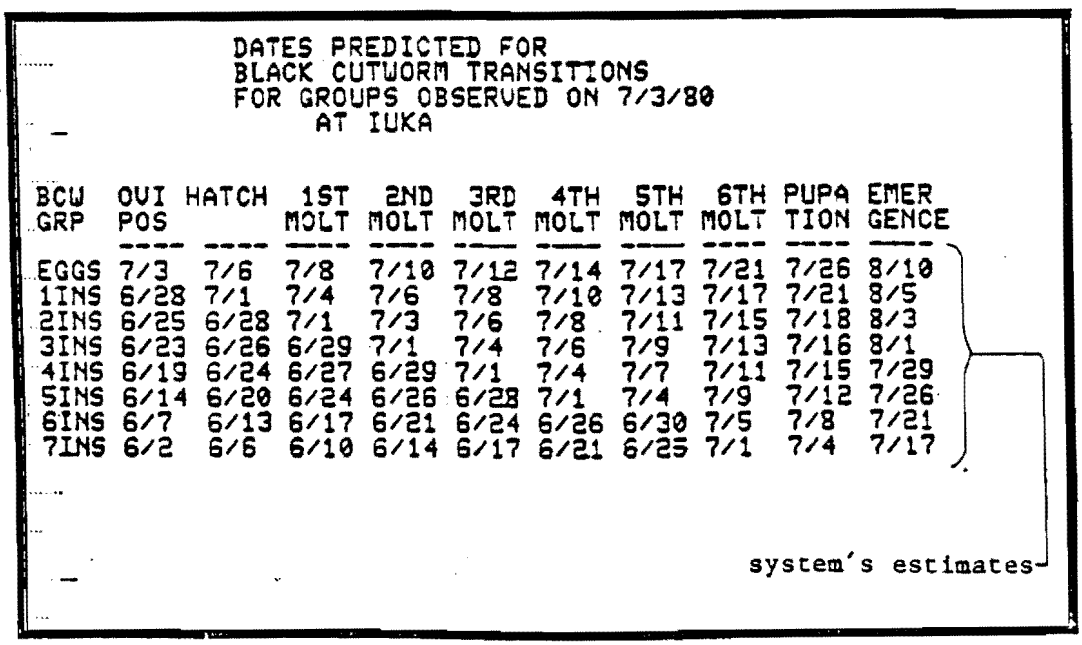


THE NUMBER OF EGGS LAID PER ACRE PER NIGHT

WEEK BEGINNING: 3-17	3-24	3-31	4-7	4-14	4-21	4-28	5-5	5-12	5-19	5-26
EGGS LAID THE REGION										
0.0	0.0	0.0	37.0	0.0	57.0	28.5	0.0	0.0	0.0	0.0
EGGS LAID IN THE FIELD										
0.0	0.0	0.0	29.6	0.0	57.0	28.5	0.0	0.0	0.0	0.0

system's estimates

TOUCH SCREEN TO CONTINUE.



DATES PREDICTED FOR
BLACK CUTWORM TRANSITIONS
FOR GROUPS OBSERVED ON 7/3/80
AT IUKA

BCW GRP	OVI POS	HATCH	1ST MOLT	2ND MOLT	3RD MOLT	4TH MOLT	5TH MOLT	6TH MOLT	PUPATION	EMERGENCE
EGGS	7/3	7/6	7/8	7/10	7/12	7/14	7/17	7/21	7/26	8/10
1INS	6/28	7/1	7/4	7/6	7/8	7/10	7/13	7/17	7/21	8/5
2INS	6/25	6/28	7/1	7/3	7/6	7/8	7/11	7/15	7/18	8/3
3INS	6/23	6/26	6/29	7/1	7/4	7/6	7/9	7/13	7/16	8/1
4INS	6/19	6/24	6/27	6/29	7/1	7/4	7/7	7/11	7/15	7/29
5INS	6/14	6/20	6/24	6/26	6/28	7/1	7/4	7/9	7/12	7/26
6INS	6/7	6/13	6/17	6/21	6/24	6/26	6/30	7/5	7/8	7/21
7INS	6/2	6/6	6/10	6/14	6/17	6/21	6/25	7/1	7/4	7/17

system's estimates

TOUCH SCREEN TO CONTINUE.

Figure 13 continued. The user is shown the inferred number of eggs laid in his field and in the region around the field in the top panel. The bottom panel displays the results of the BCW simulation model. This is in the form of a *transition* matrix which states when each of the larval stages will undergo a transition to a new stage.

LEAF STAGE OF CROP
NUMBER OF FULLY EMERGED LEAVES
(COLLAR EXPOSED)

DATE—>	1 6/29	2 7/2	3 7/5	4 7/7	5 7/10	6 7/12	7 7/14	8 7/17	9 7/19	10 7/21
INSTAR 1	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	1.0	2.0	1.0	1.0	0.0	0.0	0.0	0.0
6	4.0	0.0	0.0	0.0	2.0	3.0	3.0	1.0	0.0	0.0
7	0.0	4.0	0.0	0.0	0.0	0.0	1.0	2.0	2.0	1.0

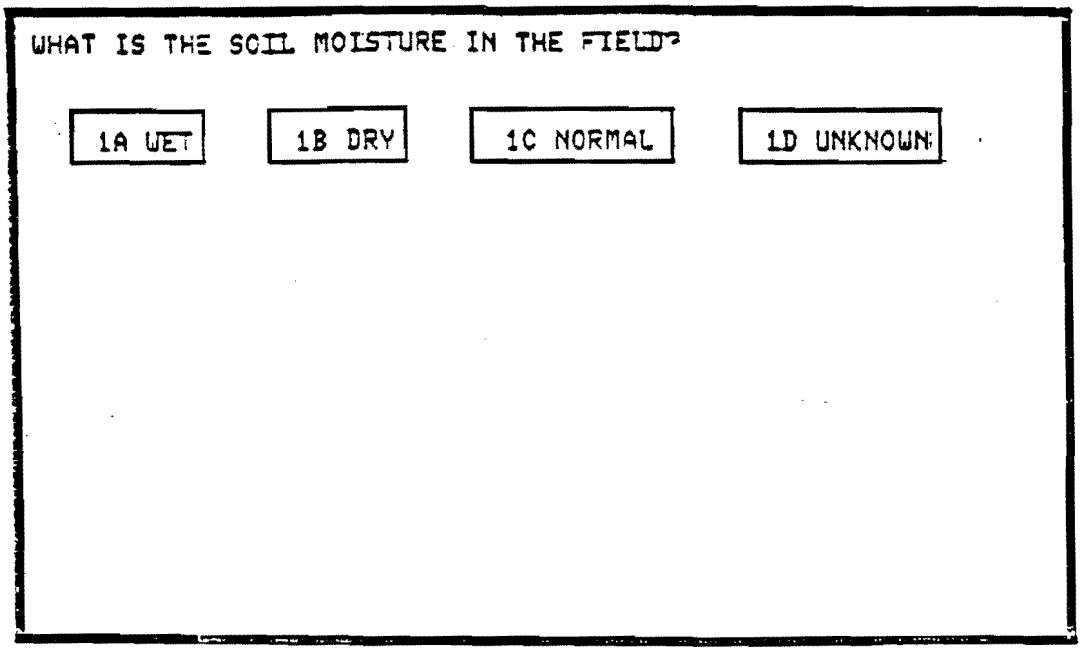
system's estimates

TOUCH SCREEN TO CONTINUE.

ARE THE WEEDS GREATER THAN 4 WEEDS/FOOT OF ROW?

1A YES 1B NO 1C UNKNOWN

Figure 13 continued. The user is shown the results of the process of putting BCW development in terms of corn development. This is in the form of a matrix that shows the population level of each of the instars approximately in the middle of the period of a leaf stage emergence and the next larger one for corn leaf stage emergences 1 through 10. In the bottom panel, the user is prompted for whether his field has a large amount of weeds. This and the next two questions will be used in estimating corn damage and yield.



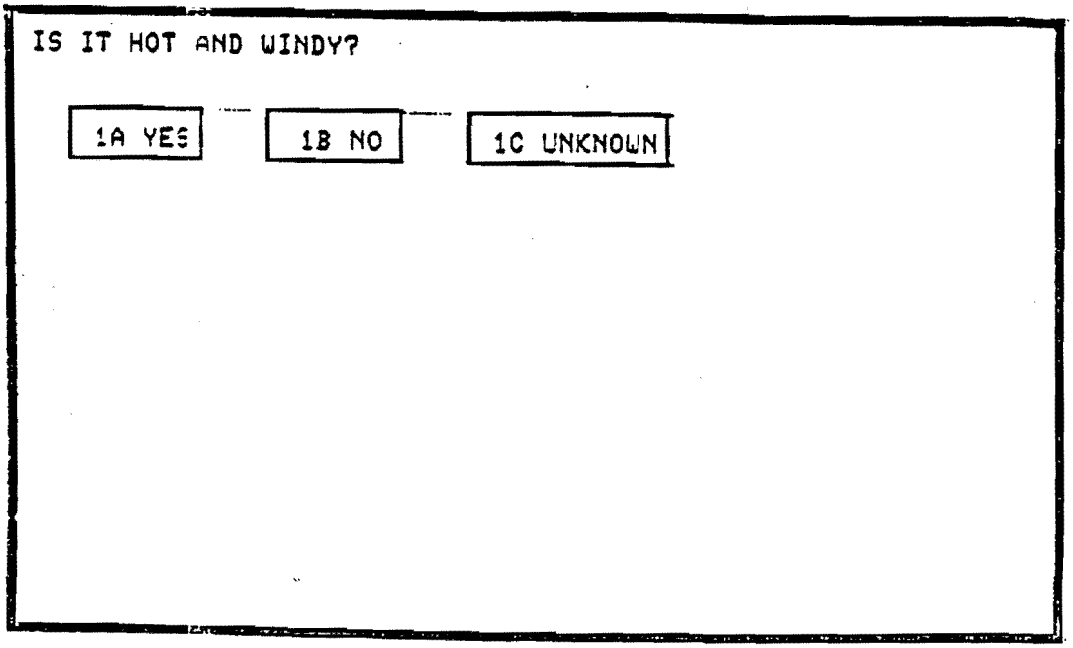


Figure 13 continued. The user is queried for the soil moisture in the field in the top panel. With certain responses to this question, the question in the bottom panel may be asked.

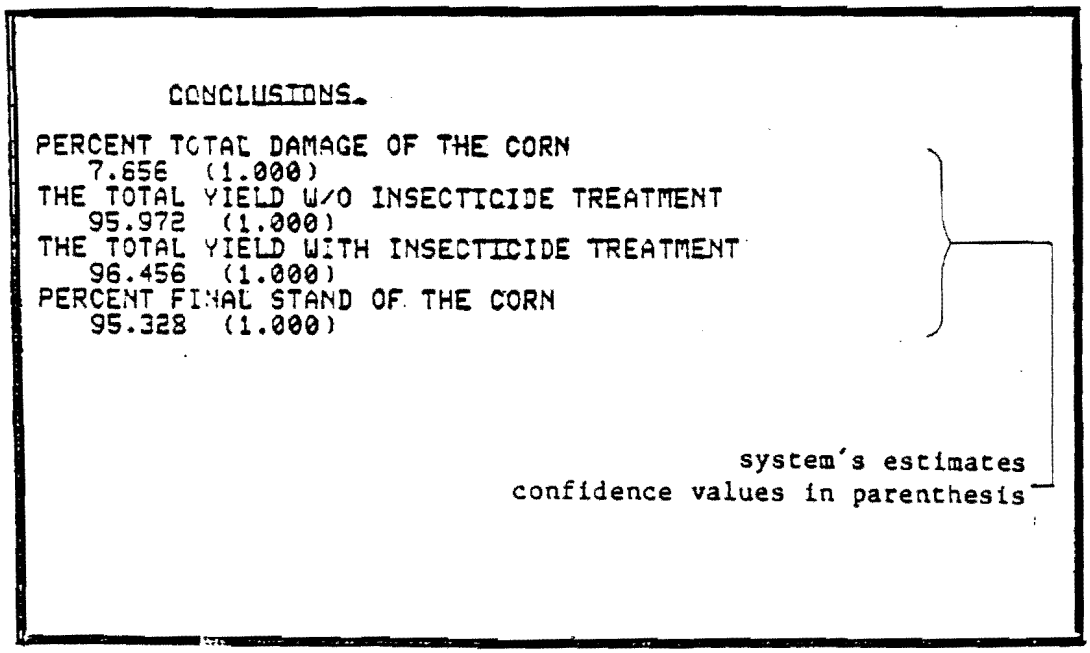


Figure 13 continued. The user is given the results of the consultation in the top panel. These are the GOAL variables of PLANT/cd. The bottom panel shows the user being asked whether he wants to continue with another session.

CHAPTER 6

VALIDATING EXPERIMENTS

This chapter describes some validating experiments using data gathered in 1982 to check the performance of PLANT/cd. These experiments were designed to test the ability of PLANT/cd to predict damage from larval counts (Experiment 1) as well as from moth trap counts (Experiment 2).

6.1. The Data

Ron Meyer, an entomologist with the Illinois State Natural History Survey, collected field data in 1982 to verify some aspects of Steve Troester's BCW simulation programs. This data included larval counts from several test areas of 14 fields in Illinois. Several of the fields were visited twice. Also collected was information such as the planting date, the corn variety, whether an insecticide was used, a description of spring weediness, tillage in the fall and spring, and previous crop. The complete list of collected items is shown in Figure 14.

The spring weediness data for the period of time before Meyer visited a field relied on the memory of the farmer or on Meyer's estimate of what it was. This was a marked weakness in the data. Also, several of the fields had areas that had undergone insecticide treatment. This would cause the predicted damage from PLANT/cd and the actual damage to differ. In these experiments, damage was measured during the growing season rather than at harvest time. Harvest time data was not available during the time these experiments were run. Hence, in some fields the corn damage was assessed when damage was occurring. This would also lead to a difference between predicted damage and the damage that was measured.

Also collected in 1982 was extensive moth trap counts for the whole of Illinois. This effort was done by Steve Briggs, Extension Specialist with the Department of Agricultural Entomology at the University of Illinois and the Natural History Survey. The set of moth trap catches in the closest county to a field

1982 BCW Data Form

VALIDATION OF BCW LOSS

Observer _____ Date _____

Information needed from grower:

1. damage history
2. fall tillage and weediness
   weed type
3. previous crop
4. spring weediness
   tillage and dates
   weed type (specific as possible)
5. planting date
6. insecticide history, this crop
7. mid or full season corn
8. plants per acre goal
9. expected yield goal, bu/A
10. if replant situation: cost of insecticide treatment
    cost to replant

Information needed from field - 1st visit:

11. i.d. must be able to get to field, but also to the damage part
12. type of land (high, low, well-drained, rolling, etc.)
13. border vegetation (to get hot spots—tree lines, creeks, high land or fence rows for wind break, clues to weeds)
14. extent of damage—whole field or spots
15. stage of growth (get to 1/10 leaf stage)
16. number of plants healthy, and cut above or below ground per 100 ft of row
17. number larvae/100 plants (get accurate instar numbers)
18. weed density (broad leaves, grasses/ft of row)
19. soil moisture (wet, normal, dry)
20. weather conditions (hot and/or windy to make soil dry)
21. repeat 16 in treated or undamaged area

Subsequent visits through 6th leaf

14-21 repeat

Harvest time visit

22. plant count in damaged and undamaged
23. bu/A in damaged and undamaged areas (100 ft row, row width)

Figure 14. The data form used in gathering the 1982 data.

was averaged and used to test the ability of PLANT/cd to predict damage from moth trap catches and the indicator of spring weediness, the attractiveness rating. There was a problem with the trap data. There were *no report* situations in which a cooperating farmer failed to send in the data for the week. This could have been because there was little or no activity for the week. Often, it looked as if the farmer placed the counts for a week with little activity with a neighboring week that had some worth reporting. For this experiment, a *no report* was counted as zero. The attractiveness rating of the fields were inferred by Meyer and the author.

6.2. The Experiments

Two experiments were done using the 1982 data collected by Meyer. The first experiment was designed to test the ability of PLANT/cd to predict BCW damage from larval counts in the field typically measured in May. The second experiment was designed to test the ability of PLANT/cd to predict damage from moth trap catches whose measurements began in the middle of March.

For the first experiment, PLANT/cd was run for 45 sets of larval counts in 14 fields. Actual temperature data up to around August 15 was used. Projected temperature data was used for the period after August 15 using the database provided by the Natural History Survey. This would not be like a real consultation which would only rely on actual temperature data up to the time of the consultation which would be much earlier. This was done to eliminate errors from predicting temperature data so that only model errors would be involved. The predicted percentage of corn cut and actual percentage of corn cut was then compared. Naturally the first experiment is easier to do than the second since it involves processes that are better known and the prediction is for a shorter time span.

The results of the first experiment are displayed in Table 1. The first column of this table gives the identification of the test site for the larval count. The first two letters in this identification are the first two characters in the last name of the farmer whose field the test site was on. The third alphabetic character (optional) indicates which field the test site was on. The following number indicates whether this was the first or second visit to this test site. The last alphabetic character is the test site letter. The test sites that are marked with an asterisk (*) were treated with insecticide. The dagger (†) indicates that this

Table 2. This table summarizes the inputs and results of experiment 1. See text for explanation of the entries.

Experiment 1

ID	Instars/100 Plants					Weather Station	Trap County	Plant Date	Obs. Date	Corn Variety	Real Cut	Proj. Cut
	3rd	4th	5th	6th	7th							
LaIA†	0.25(1)	0.00	0.00	0.21	0.00	Vincennes	Richland	4/25	5/24	Full	3%	1%
LaIB†	0.00	0.00	0.00	0.30	0.74	Vincennes	Richland	4/25	5/24	Full	4%	4%
TaIA†	0.00	0.00	0.00	0.00	0.18	Vincennes	Richland	5/10	5/24	Full	1%	0%
TaIB†	0.00	0.11	0.34	0.11	0.00	Vincennes	Richland	5/10	5/24	Full	2%	2%
TaIC†	0.00	0.75	0.38	0.75	0.00	Vincennes	Richland	5/10	5/24	Full	3%	7%
GaA1A	0.00	4.11	5.48	1.37	1.37	Iuka	Marion	4/24	5/12	Mid	22%	38%
GaA1D•	0.00	1.23	3.70	2.47	0.00	Iuka	Marion	4/24	5/12	Mid	10%	25%
GaA1C	0.00	1.41	4.23	1.41	0.00	Iuka	Marion	4/24	5/12	Mid	10%	23%
GaA1F	1.15	4.60	1.15	0.00	0.00	Iuka	Marion	4/24	5/12	Mid	7%	21%
GaA2A†	0.00	1.08	3.78	1.08	5.95	Iuka	Marion	4/24	5/17	Mid	35%	36%
GaA2B†	1.04	1.04	2.07	1.04	0.00	Iuka	Marion	4/24	5/17	Mid	18%	12%
GaA2C•	0.00	0.00	0.00	0.00	0.51	Iuka	Marion	4/24	5/17	Mid	10%	2%
GaA2D•	0.00	0.00	0.52	0.00	0.00	Iuka	Marion	4/24	5/17	Mid	3%	2%
GaA2E•	0.00	0.00	0.00	0.00	0.00	Iuka	Marion	4/24	5/17	Mid	4%	0%
GaB1A	0.56	3.09	1.12	0.00	0.00	Iuka	Marion	4/28	5/12	Mid	12%	15%
GaB1B	0.49	0.97	1.46	0.49	0.00	Iuka	Marion	4/28	5/12	Mid	7%	11%
GaB2A†	0.00	0.55	1.65	0.00	1.10	Iuka	Marion	4/28	5/17	Mid	21%	10%
GaB2B†	0.00	1.05	1.05	0.00	1.05	Iuka	Marion	4/28	5/17	Mid	21%	10%
SiIA	0.00	0.18	0.36	0.18	0.18	St Louis	St Clair	4/25	5/11	Full	4%	3%
SiIB	0.00	0.23	0.68	0.45	0.23	St Louis	St Clair	4/25	5/11	Full	4%	5%
Si2A†	0.00	0.24	0.24	0.24	0.71	St Louis	St Clair	4/25	5/17	Full	5%	5%
Si2B•	0.00	0.00	0.00	0.00	0.00	St Louis	St Clair	4/25	5/17	Full	0%	0%
BiA†	0.00	0.00	0.52	1.82	0.78	St Louis	St Clair	5/2	5/11	Full	13%	8%
BiB1A	0.84	0.42	1.26	0.00	0.00	St Louis	St Clair	5/2	5/11	Full	1%	9%
BiB1B	0.00	0.00	1.27	0.64	0.00	St Louis	St Clair	5/2	5/11	Full	2%	7%
BiB2A†	0.00	0.24	0.24	0.71	0.24	St Louis	St Clair	5/2	5/18	Full	7%	4%
BiB2B†	0.00	0.67	0.89	0.22	0.00	St Louis	St Clair	5/2	5/19	Full	5%	5%
KI†	0.00	0.33	1.16	0.17	0.17	St Louis	St Clair	4/22	5/11	Full	4%	6%
TaIA	0.00	0.60	1.51	0.30	0.00	St Louis	Jersey	5/2	5/14	Full	5%	9%
TaIB•	0.00	1.00	0.80	0.40	0.00	St Louis	Jersey	5/2	5/14	Full	9%	8%
Ta2A†	0.00	0.50	0.00	0.25	0.00	St Louis	Jersey	5/2	5/18	Full	6%	3%
Ta2B•	0.00	0.17	0.00	0.00	0.00	St Louis	Jersey	5/2	5/19	Full	4%	1%
BeIA	0.00	4.41	6.61	1.32	0.88	St Louis	Jersey	4/28	5/14	Full	69%	46%
Be2A†	0.78	3.13	3.91	7.03	3.13	St Louis	Jersey	4/28	5/19	Full	29%	56%
Be2B†	0.00	0.00	0.20	0.20	0.81	St Louis	Jersey	4/28	5/18	Full	8%	4%
SpA†	0.16	0.16	0.49	0.81	0.16	Perry	Pike	4/25	5/19	Full	10%	5%
SpB†	0.00	0.14	0.43	0.29	0.29	Perry	Pike	4/25	5/19	Full	5%	4%
TyA†	0.00	0.19	0.00	0.19	0.19	Perry	Pike	5/5	5/18	Full	4%	2%
TyB†	0.20	0.78	0.59	0.20	0.20	Perry	Pike	5/5	5/18	Full	6%	7%
KaIA	0.20	0.60	0.80	0.00	0.00	Quincy	Adams	5/2	5/13	Mid	2%	5%
KaIB	0.00	0.00	0.28	0.00	0.00	Quincy	Adams	5/2	5/13	Mid	1%	1%
Ka2A†	0.00	0.22	0.66	0.00	0.22	Quincy	Adams	5/2	5/18	Mid	4%	3%
Ka2B†	0.00	0.00	0.00	0.00	0.41	Quincy	Adams	5/2	5/19	Mid	1%	1%
BaA†	0.52	1.04	1.91	0.69	1.39	Quincy	Adams	5/8	5/18	Full	12%	17%
BaB†	0.72	1.44	1.08	0.36	0.36	Quincy	Adams	5/8	5/19	Full	9%	14%

Notes.

• *Indicates that this test area was treated with insecticide.*

(1) *These were 1st instar larvae.*

† *Indicates that this test area was used for experiment 2.*

site was also chosen for the second experiment.

The next five columns in Table 1 are the larval counts per 100 plants. The next column is the closest weather station to the field. The closest county that had moth traps is in the next column. (Moth trap counts were used in Experiment 2.) The corn planting date is next, followed by the date when the test site was observed. The next column is the corn variety.

The last two columns are the observed percentage cut corn and the percentage cut corn predicted by PLANT/cd, respectively[1]. As one can see, PLANT/cd does fairly well in this experiment in which larval counts are given. One should be cautioned in using the results from the sites treated with insecticide. If the site is treated, then one would expect higher predicted damage than actual when using counts taken before treatment. One would expect less predicted damage than actual when counts are taken while treatment was occurring and if damage occurred to any extent before treatment began.

The second experiment involved using moth traps counts and the field attractiveness ratings to predict cutworm damage. This prediction involves a longer span of time and is less accurate. Table 2 displays the inputs and results of this experiment. The first column is the field identification. The first two letters of this identification are the first two characters of the farmer's last name. The last (optional) character indicates which field if two fields of the same farmer were used. This experiment was done on a field basis since the weediness indicator was on a field basis. Each field consists of two rows of this table. The first row gives the moth trap counts of the closest county having traps (from Table 1). These counts occupy columns 3 to 13. The counts were recorded on a weekly basis from 3/14 to 5/23. As mentioned before, the trap counts from the chosen county were averaged. The second row in columns 3 to 13 are the attractiveness ratings for the field for the same weeks. These ratings were given after the fact and were therefore estimated by Meyer and the author. Using knowledge of tillage practices and a description of the weediness of the field when Meyer visited it, it was possible to infer what the weediness was before Meyer visited the field, both before and after tillage. The rest of the columns give other needed information such as the planting date and the corn variety. The last two columns show the real

[1]Percentage cut corn represents the ratio of plants that died due to BCW damage to the total planted crop. The predicted percentage cut corn is estimated when no more damage is being done to the field.

Table 3. This table summarizes the inputs and results of experiment 2. See text for explanation of the entries.

Experiment 2

Field ID	Cats. or Attr	Week Beginning											Weather Station	Plant Date	Corn Variety	Real Cat	Proj. Cat
		3/14	3/21	3/28	4/4	4/11	4/18	4/24	5/2	5/9	5/16	5/23					
La	Cats	0.0	0.0	0.0	2.0	15.3	14.3	12.3	9.3	10.6	3.6	1.3	Vincennes	5/4	Full	4%	3%
	Attr	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.1	0.1	0.1	0.1					
Ta	Cats	0.0	0.0	0.0	2.0	15.3	14.3	12.3	9.3	10.6	3.6	1.3	Vincennes	5/10	Full	2%	4%
	Attr	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.2	0.2	0.2					
GaA	Cats	0.0	0.0	8.0	4.0	7.0	2.0	1.0	0.0	0.0	0.0	0.0	Iuka	4/24	Mid	27%	2%
	Attr	0.7	0.7	0.7	0.7	0.7	0.7	0.5	0.5	0.5	0.5	0.5					
GaB	Cats	0.0	0.0	8.0	4.0	7.0	2.0	1.0	0.0	0.0	0.0	0.0	Iuka	4/28	Mid	17%	2%
	Attr	0.7	0.7	0.7	0.7	0.7	0.7	0.5	0.5	0.5	0.5	0.5					
Si	Cats	0.0	0.6	5.3	5.0	9.0	9.0	15.6	1.3	9.3	6.3	5.3	St. Louis	4/25	Full	5%	4%
	Attr	0.7	0.7	0.7	0.7	0.7	0.7	0.5	0.5	0.5	0.5	0.5					
BuA	Cats	0.0	0.6	5.3	5.0	9.0	9.0	15.6	1.3	9.3	6.3	5.3	St. Louis	5/2	Full	13%	2%
	Attr	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.2	0.2	0.2	0.2					
BuB	Cats	0.0	0.6	5.3	5.0	9.0	9.0	15.6	1.3	9.3	6.3	5.3	St. Louis	5/3	Full	6%	3%
	Attr	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.2	0.2	0.2	0.2					
Kl	Cats	0.0	0.6	5.3	5.0	9.0	9.0	15.6	1.3	9.3	6.3	5.3	St. Louis	4/22	Full	4%	3%
	Attr	0.6	0.6	0.6	0.6	0.6	0.6	0.3	0.3	0.3	0.3	0.3					
Te	Cats	0.0	0.0	16.3	0.0	17.6	2.0	12.6	4.3	3.6	1.6	3.0	St. Louis	5/3	Full	6%	2%
	Attr	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.3	0.3	0.3	0.3					
Be	Cats	0.0	0.0	16.3	0.0	17.6	2.0	12.6	4.3	3.6	1.6	3.0	St. Louis	4/28	Full	18%	1%
	Attr	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.3	0.3	0.3	0.3					
Sp	Cats	0.0	0.0	1.5	0.5	8.0	14.5	10.0	6.0	1.0	1.0	1.0	Perry	4/25	Full	8%	3%
	Attr	0.5	0.5	0.5	0.5	0.5	0.5	0.2	0.2	0.2	0.2	0.2					
Ty	Cats	0.0	0.0	1.5	0.5	8.0	14.5	10.0	6.0	1.0	1.0	1.0	Perry	5/5	Full	5%	3%
	Attr	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.3	0.3	0.3	0.3					
Ks	Cats	0.0	0.0	4.0	0.0	14.5	2.0	7.0	3.5	0.0	1.5	0.5	Quincy	5/2	Mid	3%	1%
	Attr	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.1	0.1	0.1	0.1					
Be	Cats	0.0	0.0	4.0	0.0	14.5	2.0	7.0	3.5	0.0	1.5	0.5	Quincy	5/8	Full	11%	2%
	Attr	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.2	0.2	0.2					

percentage of cut corn and the predicted estimate. The real percentage was obtained by averaging the percentage cut on the selected test sites (indicated by a † in Table 1) of a field and was measured on the observation date for the field. The observation date is given in the table of Experiment 1. These were the later visits of an untreated test site, if possible. As can be seen, the program performed poorly for some cases, especially those in which high actual damage occurred. The tendency is that PLANT/cd underestimates the real damage in such cases.

6.3. Discussion of Results

Results of Experiment 1 were much better than those of Experiment 2. To a large extent this was expected because the predictions that are done in Experiment 1 are of a shorter time span than Experiment 2. The time span for Experiment 1 was typically 2 months and the time span for Experiment 2 was 3 to 4 months. Also the processes involved in Experiment 1 are better understood than those involved in Experiment 2.

Let us discuss some of the problems that can cause the difference between the real damage and the estimated damage. There is a problem of sampling both the larval population in the field and the damaged plants. These were done by hand. There is an error of not being able to find the total larval population present in the sample area. There is the same problem for sampling the plants themselves. A plant can die due to cutworm damage and leave little trace of its existence. Plants that have undergone regrowth after damage can be misclassified. This is due to the fact that a damaged plant, if after some growth from recovering from its damage, can look like a plant that never has been damaged. Also the time when the sampling is done can lead to an error. If a damage count is done before the damage process is finished, then the predicted damage would be higher than what was counted. With these potential sources of error in mind, the results of Experiment 1 are fairly good.

If we examine the results of Experiment 2 in Table 3 we can see poor performance of PLANT/cd especially when there is real damage above a few percent. This suggests that there are some additional factors causing high damage that are not yet modelled. It is impossible for the moth population indicated by the moth trap counts to generate enough of a larval population to cause the amount of damage

observed assuming the oviposition rate in the model. This must mean that the moth population must increase or oviposition must increase beyond what is currently assumed for these high damage fields. Since the knowledge encoded in PLANT/cd is modular, it is easy to modify it when the processes that cause high damage become known.

CHAPTER 7

THE PLANT/CDP IMPLEMENTATION

This chapter describes a system, PLANT/cdp, that was designed to predict whether a field is likely to be damaged by Black Cutworms or not, rather than the degree of damage, as in PLANT/cd. PLANT/cdp is a predecessor of PLANT/cd and was developed using the EMYCIN general purpose expert system [van Melle et al. 1979]. PLANT/cdp was designed to be an initial exploration of capturing human expert knowledge in the BCW damage domain. Through talks with Chip Guse, who was my expert collaborator for the PLANT/cdp implementation, it was found that human expert knowledge was lacking in the areas of the domain that would allow for predicting extent of damage. (See Section 3.2.) The goal of my thesis work was an expert system to predict extent of damage and when we realized that the approach taken in PLANT/cdp was not going to work for predicting extent of damage, PLANT/cdp was put aside. PLANT/cdp was therefore never improved to the point of being worthy of any serious testing. However, predicting damageable fields is a reasonable thing to do, and PLANT/cdp is documented here for anybody who may wish to extend this work.

The knowledge base functions by checking whether the key elements to cutworm damage, *sufficient oviposition, larval survival,* and *synchrony* will occur for a field. Unlike PLANT/cd in which most of the expert knowledge is encoded in a simulation model, most of the expert knowledge is encoded in the expert rules themselves in PLANT/cdp. Chip Guse was able to provide a *surface* model for this problem. Most of the rules in this knowledge base deal with predicting whether the field is attractive enough for oviposition. Weediness, a critical factor to oviposition attractiveness of a field, is predicted from spring and fall tillage practices, a measure of the current weediness, and the weediness of the field in the fall. Also oviposition is assumed to be high if the field is flooded. Oviposition is assumed to relate to weediness in the manner shown in Figure 15. Notice that oviposition is less at extreme amounts of weediness.

The assumption is made that a field will escape cutworm damage if the number of Fahrenheit degree-days (FDD) between the start of oviposition and planting date is less than 90. If the number of

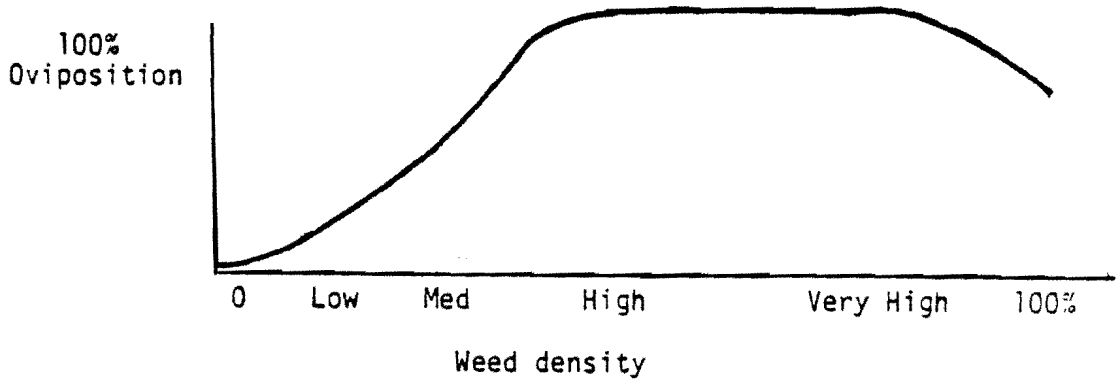


Figure 15. This figure shows the relation between weediness of the field and oviposition.

FDD is in the range 270-500, then a statement is printed out to inform the user that synchrony is at a maximum.

It is assumed that the larvae will be able to survive if the spring tillage has not been repetitive tillage or plowing since this will leave enough food supply. If the tillage has been plowing or repetitive tillage, and if the FDD is less than 360, survival occurs.

As mentioned before, this knowledge base uses EMYCIN (written in INTERLISP). The knowledge base was put on the DEC-2020 at SUMEX-AIM. The few times PLANT/cdp has been shown to experts, the opinion was that it missed the correct prediction 50% of the time. The experts mentioned that it did show promise to become more accurate.

7.1. The EMYCIN Rules

Appendix B contains a listing of the knowledge base. This includes a rule listing, a parameter listing, and a special functions listing. The rule listing contains three versions for each rule. The first version is the English paraphrasing of the rule. The second version is the terse paraphrasing, and the last version is the actual rules. The rule base consists of 25 rules. The reader is referred to [van Melle 1981] which is the users' manual for EMYCIN and this explains the terminology used in the listings. Also

references [van Melle et al. 1979] and [Shortliffe 1976] explain how EMYCIN and its ancestor, MYCIN, work in general terms.

7.2. A Session

Below is a session with PLANT/cdp:

```
4-Apr-81 09:04:19
[consultation of 9-Mar-81 3:02AM]

------FIELD-102------
1) What is the identifying name of the field under
   consideration?
** TEST2
2) What is the closest weather station to test2? Enter a?
   to see the allowed weather stations.
** BELLEVILLE
3) Has test2 had a past history of Black Cutworm damage?
** NO
4) Has test2 been flooded?
** NO
5) What is the weediness of test2 in the spring? You can
   enter ELABORATE to see elaboration on the categories of
   weediness.
** EXTENSIVE
6) Which one of the following classes of weeds
   predominates test2 in the spring: broadleaves or
   grasses?
** BROADLEAVES
7) What is the spring tillage that was done on test2?
** CHISEL
8) What is the date of spring tillage? Enter response in
   the form of: 4-Apr-81
** 10-Mar-81
9) What is the date of the first moth flight? Enter
   response in the form: 4-Apr-81 (If you don't know this
   date, answer NOTKNOWN.)
** 20-Mar-81
10) What is the fall tillage that was done on test2?
** CHISEL
11) What was the weediness of test2 in the fall? You can
    enter ELABORATE to see elaboration on the categories
    of weediness.
** PATCHY
12) Which one of the following classes of weeds
    predominated test2 in the fall: broadleaves or
    grasses?
** BROADLEAVES
I calculate the oviposition attractiveness rating to be 1.0
 (.9) or .2 (.8).
13) What is the estimated planting date? Enter response in
    this form: 4-Apr-81
```

** 25-Apr-81
Synchrony is a maximum for this case.
I calculate the 90 DD accumulation from moth flight to occur on day 114. This is compared to the estimated planting date which will occur on day 139.

Considering the information that you presented, I would say that test2 could have Black Cutworm damage.

Chip Guse agreed with the prediction above.

CHAPTER 8

SUMMARY

This thesis has described the application of the ADVISE meta-expert system to the development of a specialized expert system that predicts Black Cutworm damage (PLANT/cd). In order to build an expert system using the ADVISE inference system, one needs to:

(1) assemble knowledge about the problem by consulting with domain experts and reviewing relevant literature,

(2) encode the knowledge in the form of decision rules described in Section 2.2.2 and 3.5,[1]

(3) select the inference mechanism(s) to be used on the knowledge base, and

(4) test the knowledge base.

Through this implementation, experience was gained in building expert systems. The next two sections are some reflections on this experience.

8.1. Difficulties Encountered using ADVISE

The ADVISE project has been a challenging and novel experiment in expert system design. Some of the design goals of ADVISE led to difficulties, but the project has been quite successful over all, and some of the same design goals have contributed to this success. The following paragraphs elaborate on some design decisions and their effects on the development of ADVISE.

The choice of Pascal led to a large development time. LISP was the alternative choice. The author and another student argued for LISP as the implementation language in the initial design discussions, but they were outvoted. As a result people working on ADVISE spent much time building software tools that already existed in LISP. Pascal has no run-time interpreter and this was a continual problem. The TRAP module, the RULE PARSER, and the RULE EVALUATOR would have had a simpler implementation

[1] ADVISE also allows for the acquisition of rules through inductive inference.

with a run-time interpreter. On the other hand ther are some benefits of using Pascal as the implementation language. The system is portable and applicable to a wide spectrum of computers (and potentially microcomputers). Pascal is also more efficient than the traditional AI language, LISP. Because ADVISE is a meta-expert system, development of application systems does not require redevelopment of the functionality inherent in ADVISE. This reduces the development time of specialized expert systems and the large initial development time of ADVISE is not felt by applications of ADVISE.

Having a common set of tools to build expert systems is well worth their cost of development in the time saved in implementing different expert systems. As indicated in the above paragraph, Pascal is a language that is week in the kinds of tools needed for building expert systems. In order to avoid each module of ADVISE duplicating tools, it was decided to structure ADVISE to share common tools as much as possible. This led to reduced development time overall of the ADVISE system. At a higher level, the ADVISE modules can be thought of as tools for building expert systems. Instead of duplicating these tools for each application, ADVISE offers them prebuilt.

The system was flexible, but this led to a large and sometimes slow execution. The design goal of keeping ADVISE general contributes to this. (ADVISE now contains more than 25 thousand lines of commented Pascal.) We are now using the notion of *compiling an instance* of an expert system to reduce the overhead of of a flexible system. The hope is to put these compiled instances on microcomputer-based systems.

Building an expert system is an effort that requires more than one person. This is especially true for the case when a lot of tool building has to be done before any real work on the expert system can begin. Any expert system that is robust will require a lot of man-hours. Typically there have been 4-6 graduate students working on ADVISE at any given time. There has been more than 5 man-years of effort in the ADVISE project.

Procedures for user-friendly interaction with ADVISE requires much programmer time and are large. User-friendliness is not cheap. Much of the top level programming in the PLANT/cd and PLANT/ds (diagnosis of soybean diseases) applications of ADVISE is devoted to parsing user input, range checks on user input, help and explanation facilities, window creation, and window management. Further

enhancements such as word completion, and spelling correction will add even more overhead. There is hope that the inclusion of the "frame manager" software developed for a medical computer system which contains a robust set of tools for window creation as well as user input parsing and user input validity checks will help.

With ADVISE changes to the knowledge base are easy. The RULE PARSER makes it possible to quickly modify the knowledge base. The original version of PLANT/ds [Chilausky 1979] had no rule parser and making changes to the knowledge base was extremely difficult.

The design goal of separating tactical and strategic information is still not accomplished. Such solution is more complex than one would at first imagine. Some questions still have to be answered in representing strategic (control) information:

- What is the most appropriate language for representing strategic knowledge of a domain?
- Is the most appropriate language dependent on the domain?
- Where does strategic information go? Is it part of the rule set for a domain or is it best put in a separate knowledge structure?

Epistemological studies such as [Clancey 1981] should help in answering these questions.

8.2. Experience Gained and Difficulties Encountered from PLANT/cd

PLANT/cd gave enlightenment on the relation between deep and surface models of a domain. The Black Cutworm domain was unusual in that there were areas of the domain where there was little expert opinion. There was, however, a deep model available. This forced the expert system to be a hybrid; some knowledge was encoded in production rules and some was procedural. The production rules served to sequence and modularize the deep model (i.e., the simulation model). The advantage of such an approach is that the modules can be easily explained to the user and are also easily modified. However, the linkage between the deep and surface knowledge in PLANT/cd caused a mixing of tactics and strategy. In the PLANT/cd knowledge base, the same level of representation contains declarative non-control knowledge and control knowledge. This should be avoided, but a solution awaits to be found.

There are areas where knowledge is not robust enough to build accurate deep and/or surface models to predict BCW damage from moth trap counts. This seems to be the cause of the poor results of Experiment 2. Perhaps the goal of identifying potentially damageable fields (as was done in PLANT/cdp) is the best thing to reach for at this time. Predicting potentially damageable *areas* (e.g., whole counties) is another possibility. A major problem about Black Cutworm corn damage is the fact that the Black Cutworm is a *sporadic* pest. This makes it very hard to collect data. (It is no wonder experts have little opinion in some areas of Black Cutworm damage, if they have little data to form opinion on!)

8.3. Concluding Remarks

The prediction of extent of BCW damage has been chosen because it is important to agriculture, as its successful solution would potentially bring large economic benefits. The complexity of the problem, insufficient understanding of the BCW damage process by experts, and the lack of complete and verified multi-year data were major obstacles in this project. Despite these problems, our first small-scale attempt at solving this problem gave results that are promising and useful. To make further progress, continuing collaborative effort of entomology experts and knowledge engineers is required.

APPENDIX A

PLANT/CD RULE BASE

Below are the rule and variable listings for PLANT/cd. The format of the rules and variable declarations is described in Section 3.3 and 3.4. An English paraphrasing is included in the comments for each of the rules. Section 5.1 gives an explanation of the rules.

A.1. Rule Listing

```
CUTWORM_DAMAGE RULES
VARS = BCWVARS;

RULE1

![This rule is unconditional and is used in order to find out the
! observation date]
!
!If:     TRUE
!Then:   Get the observation date from the user and assign it to the
!        variable OBDATE.

TRUE ::> [OBDATE = TRAP(21,OBDATE,1,0)];


RULE2

![This rule is unconditional and is used in order to find out the date
! for planting corn]
!
!If:     TRUE
!Then:   Get the date for planting corn from the user and assign
!        it to the variable PLANTDATE.

TRUE ::> [PLANTDATE = TRAP(21,PLANTDATE,0,1)];


RULE3

![This rule is unconditional and is used in order to find the nearest
! weather station id]
!
!If:     TRUE
!Then:   Get the nearest weather station id from the user and assign it
!        to the variable STACODE.
```

```
TRUE ::> [STACODE = TRAP(2)];

RULE4

![This rule is unconditional and is used in order to find out the moth
! trap counts]
!
!If:     TRUE
!Then:  Get the moth trap counts from the user and assign it to the

!       variable MTRAP.

TRUE ::> [MTRAP = TRAP(3)];

RULE5

![This rule  is used in order to find out about the egg population]
!
!If:     The moth trap counts are known.
!Then:  Calculate the egg population and assign it to the variable EGGS.

[MTRAP <> $] ::> [EGGS = TRAP(6,MTRAP)];

RULE6

![This rule is unconditional and is used in order to find out the
! attractiveness rating of the field]
!
!If:     TRUE
!Then:  Get the attractiveness rating of the field from the user and
!       assign it to the variable ATTR.

TRUE ::> [ATTR = TRAP(4)];

RULE7

![This rule is used in order to find out about the egg population]
!
!If:     1) The eggs have been computed, and
!       2) the attractiveness rating of the field is known.
!Then:  Compute the eggs laid in the field and assign it to the variable
!       FEGG, and give a summary of the egg population to the user.

[EGGS <> $][ATTR <> $] ::> [FEGGS = TRAP(7,EGGS,ATTR)|TRAP(8);

RULE8

![This rule is unconditional and is used in order to find out the Black
```

```
! Cutworm population in the field]
!
!If:     TRUE
!Then:  Get the Black Cutworm population from the user and assign
!       it to the variable BCWPOP.

TRUE ::> [BCWPOP = TRAP(5)];


RULE9

![This rule is used in order to find out about the degree day table]
!
!If:     1) The observation date is known, and
!       2) The weather station id is known.
!Then:  Compute the degree day table and assign it to the variable DD.

[OBDATE <> $][STACODE <> $] ::> [DD = TRAP(12,OBDATE,STACODE)];


RULE10

![This rule is used in order to find out the planting date of the corn]
!
!If:     1) The degree day table is known,
!       2) The observation date is known,
!       3) The fractional leafstage of the corn is known, and
!       4) The corn variety is known
!Then:  Compute the planting date of the corn and assign it to the
!       variable PLANTDATE.

[DD <> $][OBDATE <> $][FRACT <> $][VARIETY <> $] ::>
  [PLANTDATE = TRAP(13,OBDATE,FRACT,VARIETY,DD)];


RULE11

![This rule is used in order to find out the corn development]
!
!If:     1) The degree day is known,
!       2) The planting date of the corn is known, and
!       3) The variety of the corn is known
!Then:  Simulate corn development and assign the results to the
!       variable CD, and display the results to the user.

[DD <> $][PLANTDATE <> $][VARIETY <> $] ::>
  [CD = TRAP(14,PLANTDATE,VARIETY,DD)]TRAP(10);


RULE12

![This rule is tried in order to find out about Black Cutworm
! development]
!
```

```
!If:     1) The degree day table is known,
!        2) The observation date < planting date, and
!        3) The egg population in the field is known
!Then:  Simulate BCW development and assign the results to the
!        variable BCWD and display the results to the user.

[DD <> $][OBDATE < PLANTDATE][FEGGS <> $] ::>
  [BCWD = TRAP(15,OBDATE,PLANTDATE,FEGGS,DD)]TRAP(9,OBDATE);


RULE13

![This rule is tried in order to find out about Black Cutworm
! development]
!
!If:     1) The degree day table is known,
!        2) The observation date is ≥ the planting date,
!        3) The Black Cutworm population is known
!Then:  Simulate BCW development and assign the results to the
!        variable BCWD and display the results to the user.

[DD <> $][OBDATE >= PLANTDATE][BCWPOP <> $] ::>
  [BCWD = TRAP(15,OBDATE,PLANTDATE,BCWPOP,DD)]TRAP(9,OBDATE);


RULE14

![This rule is used in order to find out the Black Cutworm vs leafstage
! table]
!
!If:     1) The corn development is known, and
!        2) The Black Cutworm development is known
!Then:  Compute the Black Cutworm vs. leafstage table and assign it to
!       the variable POPVSSTAGE and display the Black Cutworm vs.
!       leafstage table.

[CD <> $][BCWD <> $] ::> [POPVSSTAGE = TRAP(16,CD,BCWD)]TRAP(11);


RULE15

![This rule is used to find out the corn yield without insecticide
! treatment]
!
!If:     1) The Black Cutworm vs leafstage table has been computed,
!        2) Whether there are greater than 4 weeds/foot of row is known,
!        3) The corn variety is known, and
!        4) The soil moisture in the field is known
!Then:  Compute the corn yeild without insecticide treatment and assign
!        it to the variable YIELD1.

[POPVSSTAGE <> $][MUCHWEEDS <> $][VARIETY <> $][MOISTURE <> $] ::>
  [YIELD1 = TRAP(17,MUCHWEEDS,VARIETY,MOISTURE,0.25,CORNDAMAGE,POPVSSTAGE)];
```

RULE16

```
![This rule is used in order to find out the fractional leafstage of the
! corn]
!
!If:     1) The planting date is defined, and
!       2) The observation date is ≤ the planting date
!Then:  The fractional leafstage is = 0.0.

[PLANTDATE <> ?][OBDATE <= PLANTDATE] ::> [FRACT = 0.0];
```

RULE17

```
![This rule is used to find out the corn yield with insecticide
! treatment]
!
!If:     1) The Black Cutworm vs leafstage table has been computed,
!       2) The fractional leafstage is known, and
!       3) The soil moisture in the field is known
!Then:  Compute the corn yield with insecticide treatment and assign it
!      to the variable YIELD2.

[POPVSSTAGE <> $][FRACT <> $][MOISTURE <> $] ::>
  [YIELD2 = TRAP(18,FRACT,MOISTURE,POPVSSTAGE)];
```

RULE18

```
![This rule is used in order to find out whether it is hot and windy]
!
!If:     The soil moisture is dry.
!Then:  It is not hot and windy.

[MOISTURE = DRY] ::> [HOTWINDY = NO];
```

RULE19

```
![This rule is used in order to find out corn yield increment due to
! recovery without insecticide treatment]
!If:     1) The Black Cutworm vs leafstage table is known,
!       2) The soil moisture is known, and
!       3) Whether it is hot and windy is known.
!Then:  Compute the corn yield increment due to recovery without
!       insecticide treatment and assign it the variable YIELDINC1

[POPVSSTAGE <> $][MOISTURE <> $][HOTWINDY <> $] ::>
  [YIELDINC1 = TRAP(19,MOISTURE,HOTWINDY,TOTRECOVER,POPVSSTAGE)];
```

RULE20

```
![This rule is used in order to find out corn yield increment due to
```

```
! recovery with insecticide treatment]
!
!If:     1) The Black Cutworm vs leafstage table is known,
!       2) The fractional leafstage of the corn is known, and
!       3) The moisture of the soil is known.
!Then:  Compute the corn yield increment due to recovery with
!       insecticide treatment and assign it the variable YIELDINC2.

[POPVSSTAGE <> $][FRACT <> $][MOISTURE <> $] ::>
  [YIELDINC2 = TRAP(20,FRACT,MOISTURE,POPVSSTAGE)];


RULE21

![This rule is used in order to find the total corn yield without
! insecticide treatment]
!
!If:     1) The corn yield without insecticide treatment is known, and
!       2) The corn yield increment due to recovery without insecticide
!          treatment is known
!Then:  The total corn yield without insecticide treatment is the corn
!       yield without insecticide treatment plus the corn yield
!       increment due to recovery without insecticide treatment  and
!       assign it to the variable TYIELD1.

[YIELD1 <> $][YIELDINC1 <> $] ::>
  [TYIELD1 = YIELD1+ YIELDINC1];


RULE22

![This rule is used in order to find the total corn yield with
! insecticide treatment]
!
!If:     1) The corn yield with insecticide treatment is known, and
!       2) The corn yield increment due to recovery with insecticide
!          treatment is known.
!Then:  The total corn yield with insecticide treatment is the corn
!       yield with insecticide treatment plus the corn yield increment
!       due recovery with insecticide treatment.

[YIELD2 <>$][YIELDINC2 <> $] ::>
  [TYIELD2 = YIELD2+ YIELDINC2];


RULE23

![This rule is used in order to find the percent final stand of the corn]
!
!If:     1) The percent total damage of the corn is known, and
!       2) The percent total recovery of the corn is known
!Then:  The percent final stand of the corn is 100.0% minus the
!       difference between the percent total damage of the corn and the
!       percent total recovery of the corn.
```

```
[CORNDAMAGE <> $][TOTRECOVER <> $] ::>
  [STAND = 100.0 - (CORNDAMAGE-TOTRECOVER)];


RULE24

![This rule is used to find the fractional leafstage]
!
!If:     The observation date is > the planting date.
!Then:  Compute the fractional leafstage and assign it to the
!       variable FRACT.

[OBDATE > PLANTDATE] ::>
  [FRACT = TRAP(22,PLANTDATE,OBDATE)];


RULE25

![This rule is used to find whether there are greater than 4 weeds/foot
! of row]
!
!If:     The observation date is ≤ the planting date.
!Then:  There are not greater than 4 weeds/foot of row.

[OBDATE <= PLANTDATE] ::>
  [MUCHWEEDS = NO];
```

A.2. Variable Listing

```
BCWVARS VARS

OBDATE INTERVAL = 1..366
PROP = TRANS
THE DATE FOR WHICH THE FIELD IS OBSERVED
%%
PROP = PROMPT
ON WHAT DATE WAS THE FIELD OBSERVED?
%%
PROP = LABDATA
T
%%
;

PLANTDATE INTERVAL = 1..366
PROP = TRANS
THE DATE FOR PLANTING CORN
%%
PROP = PROMPT
WHAT IS THE PLANTING DATE?
%%
PROP = LABDATA
T
```

```
%%
;
STACODE INTERVAL = 1000..9999
PROP = TRANS
THE STATION CODE
%%
PROP = LABDATA
T
%%
;
MTRAP NOMINAL = (YES NO)
PROP = TRANS
MOTH TRAP COUNTS HAVE BEEN GOTTEN
%%
;

EGGS NOMINAL = (YES NO)
PROP = TRANS
EGGS HAVE BEEN COMPUTED
%%
;

FEGGS NOMINAL = (YES NO)
PROP = TRANS
EGGS IN THE FIELD HAVE BEEN COMPUTED
%%
;

ATTR NOMINAL = (YES NO)
PROP = TRANS
ATTRACTIVENESS RATINGS OF THE FIELD HAVE BEEN GOTTEN
%%
;

BCWPOP NOMINAL = (YES NO)
PROP = TRANS
BLACK CUTWORM LARVAL COUNTS HAVE BEEN GOTTEN
%%
;

DD NOMINAL = (YES NO)
PROP = TRANS
DEGREE-DAY TABLE HAS BEEN READ IN
%%
;

CD NOMINAL = (YES NO)
PROP = TRANS
CORN DEVELOPMENT HAS BEEN SIMULATED
%%
;
```

```
BCWD NOMINAL = (YES NO)
PROP = TRANS
BLACK CUTWORM DEVELOPMENT HAS BEEN SIMULATED
%%
;

POPVSSTAGE NOMINAL = (YES NO)
PROP = TRANS
THE BLACK CUTWORM VS LEAFSTAGE TABLE HAS BEEN COMPUTED
%%
;

FRACT INTERVAL = 0.0 .. 10.0
PROP = PROMPT
WHAT IS THE OBSERVED FRACTIONAL LEAF STAGE OF THE CORN ?
%%
PROP = TRANS
THE OBSERVED FRACTIONAL LEAF STAGE OF THE CORN
%%
;

VARIETY NOMINAL = (FULLSEASON MIDSEASON)
PROP = PROMPT
WHAT TYPE OF CORN IS BEING PLANTED?
%%
PROP = TRANS
THE CORN VARIETY
%%
PROP = LABDATA
T
%%
PROP = NOUNKNOWN
T
%%
;

MOISTURE NOMINAL = (WET DRY NORMAL)
PROP = PROMPT
WHAT IS THE SOIL MOISTURE IN THE FIELD?
%%
PROP = TRANS
THE SOIL MOISTURE IN THE FIELD
%%
PROP = NOUNKNOWN
T
%%
;
```

```
HOTWINDY NOMINAL = (YES NO)
PROP = PROMPT
IS IT HOT AND WINDY?
%%
PROP = TRANS
IT IS HOT AND WINDY
%%
PROP = NOUNKNOWN
T
%%
;
MUCHWEEDS NOMINAL = (YES NO)
PROP = PROMPT
ARE THE WEEDS GREATER THAN
4 WEEDS/FOOT OF ROW?
%%
PROP = TRANS
THERE ARE GREATER THAN 4 WEEDS/FOOT OF ROW
%%
PROP = NOUNKNOWN
T
%%
;

YIELD1 INTERVAL = 0.0 .. 100.0
PROP = TRANS
THE YIELD W/O INSECTICIDE TREATMENT
%%
;

YIELD2 INTERVAL = 0.0 .. 100.0
PROP = TRANS
THE YIELD WITH INSECTICIDE TREATMENT
%%
;

YIELDINC1 INTERVAL = 0.0 .. 100.0
PROP = TRANS
THE INCREASE IN YIELD DUE TO RECOVERY W/O INSECTICIDE TREATMENT
%%
;

YIELDINC2 INTERVAL = 0.0 .. 100.0
PROP = TRANS
THE INCREASE IN YIELD DUE TO RECOVERY WITH INSECTICIDE TREATMENT
%%
;

TYIELD1 INTERVAL = 0.0 .. 100.0
PROP = TRANS
THE TOTAL YIELD W/O INSECTICIDE TREATMENT
%%
;
```

```
TYIELD2 INTERVAL = 0.0 .. 100.0
PROP = TRANS
THE TOTAL YIELD WITH INSECTICIDE TREATMENT
%%
;
CORNDAMAGE  INTERVAL = 0.0 .. 100.0
PROP = TRANS
PERCENT TOTAL DAMAGE OF THE CORN
%%
;
TOTRECOVER  INTERVAL = 0.0 .. 100.0
PROP = TRANS
PERCENT TOTAL RECOVERY OF THE CORN
%%
;

STAND INTERVAL = 0.0 .. 100.0
PROP = TRANS
PERCENT FINAL STAND OF THE CORN
%%
;
```

APPENDIX B

PLANT/CDP RULE BASE

This appendix contains listings of the rule, parameter, special functions, and properties of some variables of the PLANT/cdp rule base that was implemented in EMYCIN. PLANT/cdp is described in Chapter 7. See [van Melle et al. 1981] for format details.

B.1. Rule Listing

This listing contains the English, terse, and LISP form of the PLANT/cdp rules.

RULE001

English form of the rule:

[This rule is tried in order to find out about whether considering the information that you presented, I would say that the field could have Black Cutworm damage]

If: 1) Cutworm eggs have been laid to a sufficient degree to cause damage,
2) Cutworm larvae are able to survive on the present weed population, and
3) Cutworm development is synchronized with corn development
Then: It is definite (1.0) that considering the information that you presented, I would say that the field could have Black Cutworm damage

Terse form of the rule:

If: 1) Ovipos,
2) Survive, and
3) Synchrony
Then: Damage = yes (1.0)

LISP form:

```
PREMISE: ($AND (SAME CNTXT OVIPOS)
               (SAME CNTXT SURVIVE)
               (SAME CNTXT SYNCHRONY))
ACTION:  (CONCLUDE CNTXT DAMAGE YES TALLY 1000)
```

where
$AND is the variable-valued logic AND of EMYCIN,

SAME is the LISP function that compares a EMYCIN PARAMETER to a value and returns a truth status, and

CONCLUDE updates the value of an EMYCIN PARAMETER with an associated certainty.

RULE002

[This rule is tried in order to find out about whether cutworm eggs have been laid to a sufficient degree to cause damage]

If: 1) Flooding has occurred to cause favorable conditions for sufficient oviposition to cause cutworm damage, or
2) The oviposition attractiveness rating for the weeds in the field is great enough to indicate a Black Cutworm problem
Then: It is definite (1.0) that cutworm eggs have been laid to a sufficient degree to cause damage

If: Flood or Weedovipossuff
Then: Ovipos = yes (1.0)

PREMISE: ($AND ($OR (SAME CNTXT FLOOD YES)
(SAME CNTXT WEEDOVIPOSSUFF YES)))
ACTION: (CONCLUDE CNTXT OVIPOS YES TALLY 1000)

RULE004

[This rule is tried in order to find out about whether cutworm development is synchronized with corn development]

If: 1) You have displayed the header: I calculate the 90 DD accumulation from moth flight to occur on day <the Julian date that corresponds to 90 FDD that is based on mfd>. This is compared to the estimated planting date which will occur on day <plantdate>., and
2) The estimated planting date is greater than or equal to the Julian date that corresponds to 90 FDD that is based on the date of the first moth flight
Then: It is definite (1.0) that cutworm development is synchronized with corn development

If: 1) You have displayed the header: I calculate the 90 DD accumulation from moth flight to occur on day <the Julian date that corresponds to 90 FDD that is based on mfd>. This is compared to the estimated planting date which will occur on day <plantdate>., and
2) Plantdate >= the Julian date that corresponds to 90 FDD that is based on mfd
Then: Synchrony = yes (1.0)

```
PREMISE: [$AND (HEADER (CDDR (TEXT NIL
"I calculate the 90 DD accumulation from moth flight to occur on day "
                        (FDDTOJDATE 90 (VAL1 CNTXT MFD))

". This is compared
to the estimated planting date which will occur on day "
                        (VAL1 CNTXT PLANTDATE)
                        ".")))
            (GREATEQ* (VAL1 CNTXT PLANTDATE)
                      (FDDTOJDATE 90 (VAL1 CNTXT MFD]
ACTION: (CONCLUDE CNTXT SYNCHRONY YES TALLY 1000)
```

RULE005

[This rule is definitional, is tried when information is received about the estimated planting date]

If: 1) The estimated planting date is greater than or equal to the Julian date that corresponds to 270 FDD that is based on the date of the first moth flight, and
2) The estimated planting date is less than or equal to the Julian date that corresponds to 500 FDD that is based on the date of the first moth flight
Then: Display the comment: Synchrony is a maximum for this case.

If: 1) Plantdate >= the Julian date that corresponds to 270 FDD that is based on mfd, and
2) Plantdate <= the Julian date that corresponds to 500 FDD that is based on mfd
Then: Display the comment: Synchrony is a maximum for this case.

```
PREMISE: [$AND (GREATEQ* (VAL1 CNTXT PLANTDATE)
                         (FDDTOJDATE 270 (VAL1 CNTXT MFD)))
               (LESSEQ* (VAL1 CNTXT PLANTDATE)
                        (FDDTOJDATE 500 (VAL1 CNTXT MFD]
ACTION: (SPRINTT "Synchrony is a maximum for this case." 0 5)
```

[FIELDRULES/antecedent]

RULE006

[This rule is tried in order to find out about whether cutworm larvae are able to survive on the present weed population]

If: 1) The spring tillage that was done on the field is one of: none disc chisel field_cultivator other, or
2) A: There is the date of spring tillage,
B: You have displayed the header: I calculate the 360 DD survival time to be <the Julian date that corresponds to

360 FDD that is based on springtilldate> from the beginning of the year. This is compared to the planting date being: <plantdate>, and

C: The estimated planting date is less than the Julian date that corresponds to 360 FDD that is based on the date of spring tillage

Then: It is definite (1.0) that cutworm larvae are able to survive on the present weed population

If: Spring_till = {none disc chisel field_cultivator other} or
A: Thereis springtilldate,
B: You have displayed the header: I calculate the 360 DD survival time to be <the Julian date that corresponds to 360 FDD that is based on springtilldate> from the beginning of the year. This is compared to the planting date being: <plantdate>, and
C: Plantdate < the Julian date that corresponds to 360 FDD that is based on springtilldate
Then: Survive = yes (1.0)

```
PREMISE: [$AND
        ($OR (SAME CNTXT SPRING_TILL
                (ONEOF NONE DISC CHISEL FIELD_CULTIVATOR OTHER)
                )
            ($AND (ONCEKNOWN CNTXT SPRINGTILLDATE)
                [HEADER (CDDR (TEXT NIL
                "I calculate the 360 DD
survival time to be"
                                (FDDTOJDATE 360
                                    (VAL1 CNTXT
                                SPRINGTILLDATE))

" from the beginning of the year. This is compared to the planting date being: "
                                (VAL1 CNTXT PLANTDATE]
                (LESSP* (VAL1 CNTXT PLANTDATE)
                    (FDDTOJDATE 360 (VAL1 CNTXT
                                SPRINGTILLDATE]
 ACTION: (CONCLUDE CNTXT SURVIVE YES TALLY 1000)
```

RULE007

[This rule is definitional, is tried when information is received about whether the field has had a past history of Black Cutworm damage]

If: The field has had a past history of Black Cutworm damage
Then: There is suggestive evidence (.5) that considering the information that you presented, I would say that the field could have Black Cutworm damage

If: Past-history
Then: Damage = yes (.5)

PREMISE: ($AND (SAME CNTXT PAST-HISTORY))
ACTION: (CONCLUDE CNTXT DAMAGE YES TALLY 500)

[FIELDRULES/antecedent]

RULE012

[This rule is tried in order to find out about whether flooding has occurred to cause favorable conditions for sufficient oviposition to cause cutworm damage]

If: 1) The field has been flooded, and
2) The date that flooding receded is greater than the date of the first moth flight
Then: It is definite (1.0) that flooding has occurred to cause favorable conditions for sufficient oviposition to cause cutworm damage

If: 1) Fldfld, and
2) Recededate > mfd
Then: Flood = yes (1.0)

PREMISE: ($AND (SAME CNTXT FLDFLD)
(GREATERP* (VAL1 CNTXT RECEDEDATE)
(VAL1 CNTXT MFD)))
ACTION: (CONCLUDE CNTXT FLOOD YES TALLY 1000)

RULE013

[This rule is tried in order to find out about whether the oviposition attractiveness rating for the weeds in the field is great enough to indicate a Black Cutworm problem]

If: 1) A: It is known whether I have considered spring weediness, or
B: An attempt has been made to deduce whether I have considered spring tillage,
2) You have displayed the header: I calculate the oviposition attractiveness rating to be <springoviposind>., and
3) The oviposition attractiveness rating of the field after spring tillage is greater than .1
Then: It is definite (1.0) that the oviposition attractiveness rating for the weeds in the field is great enough to indicate a Black Cutworm problem

If: 1) Springweediness1 is known or Spring_till1 is traced,
2) You have displayed the header: I calculate the oviposition attractiveness rating to be <springoviposind>., and
3) Springoviposind > .1
Then: Weedovipossuff = yes (1.0)

```
PREMISE: ($AND ($OR (KNOW CNTXT SPRINGWEEDINESS1)
                    (ONCEKNOWN CNTXT SPRING_TILL1 T))
               (HEADER (CDDR (TEXT NIL
        "I calculate the oviposition
attractiveness rating to be "
                              (VAL CNTXT SPRINGOVIPOSIND)
                              ".")))
               (GREATERP* (VAL1 CNTXT SPRINGOVIPOSIND)
                          .1))
ACTION: (CONCLUDE CNTXT WEEDOVIPOSSUFF YES TALLY 1000)
```

RULE014

[This rule is tried in order to find out about the oviposition attractiveness rating of the field after spring tillage or whether I have considered spring weediness]

If: 1) The weediness of the field in the spring is known, and
2) The predominant weed type in the field in the spring is known
Then: 1) There is strongly suggestive evidence (.9) that the oviposition attractiveness rating of the field after spring tillage is the conversion of the weediness of the field in the spring considering the predominant weed type in the field in the spring to the oviposition attractiveness rating, and
2) It is definite (1.0) that I have considered spring weediness

If: 1) Springweediness is known, and
2) Springpredweedtyp is known
Then: 1) Springoviposind = the conversion of springweediness considering springpredweedtyp to the oviposition attractiveness rating (.9), and
2) Springweediness1 = yes (1.0)

```
PREMISE: ($AND (KNOWN CNTXT SPRINGWEEDINESS)
               (KNOWN CNTXT SPRINGPREDWEEDTYP))
ACTION: (DO-ALL (CONCLUDE CNTXT SPRINGOVIPOSIND
                          (WEEDTOOVIPOSIND (VAL1 CNTXT
                                                 SPRINGWEEDINESS)
                                           (VAL1 CNTXT
                                                 SPRINGPREDWEEDTYP))
                          TALLY 900)
```

(CONCLUDE CNTXT SPRINGWEEDINESS1 YES TALLY 1000))

RULE015

[This rule is tried in order to find out about the oviposition attractiveness rating of the field after spring tillage or whether I have considered spring tillage]

If: The equivalent tillage that was done on the field in the spring considering tillage date compared to moth flight date is moldboard_plow
Then: 1) There is strongly suggestive evidence (.8) that the oviposition attractiveness rating of the field after spring tillage is 0, and
2) It is definite (1.0) that I have considered spring tillage

If: Tillage = moldboard_plow
Then: 1) Springoviposind = 0.0 (.8), and
2) Spring_till1 = yes (1.0)

PREMISE: ($AND (SAME CNTXT TILLAGE MOLDBOARD_PLOW))
ACTION: (DO-ALL (CONCLUDE CNTXT SPRINGOVIPOSIND 0.0 TALLY 800)
(CONCLUDE CNTXT SPRING_TILL1 YES TALLY 1000))

RULE016

[This rule is tried in order to find out about the oviposition attractiveness rating of the field after spring tillage]

If: 1) The equivalent tillage that was done on the field in the spring considering tillage date compared to moth flight date is one of: disc chisel field_cultivator other,
2) An attempt has been made to deduce whether I have considered tillage in the fall, and
3) There is the oviposition attractiveness rating of the field in the fall after fall tillage
Then: The oviposition attractiveness rating of the field after spring tillage is as follows:
If the oviposition attractiveness rating of the field in the fall after fall tillage is:
a) less than .4 then: 0.0 (.8),
b) greater or equal to .4 then: the oviposition attractiveness rating times 0.5 (.8);

If: 1) Tillage = {disc chisel field_cultivator other},
2) Fall_till1 is traced, and
3) Thereis falloviposind
Then: Springoviposind is as follows:

If falloviposind is:
a) less than .4 then: 0.0 (.8),
b) greater or equal to .4 then: the oviposition attractiveness rating times 0.5 (.8);

```
PREMISE: ($AND (SAME CNTXT TILLAGE (ONEOF DISC CHISEL
                                    FIELD_CULTIVATOR OTHER))
               (ONCEKNOWN CNTXT FALL_TILL1 T)
               (ONCEKNOWN CNTXT FALLOVIPOSIND))
ACTION:  [CONCLUDET CNTXT (VAL1 CNTXT FALLOVIPOSIND)
                  '((LT .4 800 0)
                   (GE .4 0 800))
                  TALLY SPRINGOVIPOSIND
                  (LIST '0.0 (FTIMES .5 (VAL1 CNTXT FALLOVIPOSIND]
```

RULE017

[This rule is tried in order to find out about the oviposition attractiveness rating of the field after spring tillage]

If: 1) The equivalent tillage that was done on the field in the spring considering tillage date compared to moth flight date is none,
2) An attempt has been made to deduce whether I have considered tillage in the fall, and
3) The oviposition attractiveness rating of the field in the fall after fall tillage is known
Then: There is strongly suggestive evidence (.8) that the oviposition attractiveness rating of the field after spring tillage is 1.1 times [the oviposition attractiveness rating of the field in the fall after fall tillage plus .2]

If: 1) Tillage = none,
2) Fall_till1 is traced, and
3) Falloviposind is known
Then: Springoviposind = 1.1 * (falloviposind + .2) (.8)

```
PREMISE: ($AND (SAME CNTXT TILLAGE NONE)
               (ONCEKNOWN CNTXT FALL_TILL1 T)
               (KNOWN CNTXT FALLOVIPOSIND))
ACTION:  (CONCLUDE CNTXT SPRINGOVIPOSIND
                   (TIMES 1.1 (PLUS (VAL1 CNTXT FALLOVIPOSIND)
                                    .2))
                   TALLY 800)
```

RULE018

[This rule is tried in order to find out about the oviposition attractiveness rating of the field in the fall after fall tillage or whether I have considered tillage in the fall]

If: The fall tillage that was done on the field is moldboard_plow
Then: 1) There is strongly suggestive evidence (.9) that the oviposition attractiveness rating of the field in the fall after fall tillage is 0, and
2) It is definite (1.0) that I have considered tillage in the fall

If: Fall_till = moldboard_plow
Then: 1) Falloviposind = 0.0 (.9), and
2) Fall_till1 = yes (1.0)

PREMISE: ($AND (SAME CNTXT FALL_TILL MOLDBOARD_PLOW))
ACTION: (DO-ALL (CONCLUDE CNTXT FALLOVIPOSIND 0.0 TALLY 900)
(CONCLUDE CNTXT FALL_TILL1 YES TALLY 1000))

RULE019

[This rule is tried in order to find out about the oviposition attractiveness rating of the field in the fall after fall tillage]

If: 1) The fall tillage that was done on the field is none,
2) An attempt has been made to deduce whether the weediness in the field in the fall before fall tillage is known, and
3) The oviposition attractiveness rating for the field in the fall before any tillage is known
Then: There is strongly suggestive evidence (.9) that the oviposition attractiveness rating of the field in the fall after fall tillage is the oviposition attractiveness rating for the field in the fall before any tillage plus .1

If: 1) Fall_till = none,
2) Fallweedknown is traced, and
3) Oviposind is known
Then: Falloviposind = oviposind + .1 (.9)

PREMISE: ($AND (SAME CNTXT FALL_TILL NONE)
(ONCEKNOWN CNTXT FALLWEEDKNOWN T)
(KNOWN CNTXT OVIPOSIND))
ACTION: (CONCLUDE CNTXT FALLOVIPOSIND (PLUS (VAL1 CNTXT OVIPOSIND)
.1)
TALLY 900)

RULE020

[This rule is tried in order to find out about the oviposition attractiveness rating of the field in the fall after fall tillage]

If: 1) The fall tillage that was done on the field is one of: disc chisel field_cultivator other,
2) An attempt has been made to deduce whether the weediness in the field in the fall before fall tillage is known, and
3) There is the oviposition attractiveness rating for the field in the fall before any tillage
Then: The oviposition attractiveness rating of the field in the fall after fall tillage is as follows:
If the oviposition attractiveness rating for the field in the fall before any tillage is:
a) less than .4 then: 0.0 (.8),
b) greater or equal to .4 then: the oviposition attractiveness rating times 0.5 (.8);

If: 1) Fall_till = {disc chisel field_cultivator other},
2) Fallweedknown is traced, and
3) Thereis oviposind
Then: Falloviposind is as follows:
If oviposind is:
a) less than .4 then: 0.0 (.8),
b) greater or equal to .4 then: the oviposition attractiveness rating times 0.5 (.8);

```
PREMISE: ($AND (SAME CNTXT FALL_TILL (ONEOF DISC CHISEL
                                      FIELD_CULTIVATOR OTHER))
               (ONCEKNOWN CNTXT FALLWEEDKNOWN T)
               (ONCEKNOWN CNTXT OVIPOSIND))
ACTION:  [CONCLUDET CNTXT (VAL1 CNTXT OVIPOSIND)
                  '((LT .4 800 0)
                   (GE .4 0 800))
                  TALLY FALLOVIPOSIND
                  (LIST '0.0 (FTIMES .5 (VAL1 CNTXT OVIPOSIND]
```

RULE021

[This rule is tried in order to find out about the oviposition attractiveness rating for the field in the fall before any tillage or whether the weediness in the field in the fall before fall tillage is known]

If: 1) The weediness of the field in the fall is known, and
2) The predominant weed type in the field in the fall is known
Then: 1) It is definite (1.0) that the oviposition attractiveness rating for the field in the fall before any tillage is the conversion of the weediness of the field in the fall

considering the predominant weed type in the field in the fall to the oviposition attractiveness rating, and
2) It is definite (1.0) that the weediness in the field in the fall before fall tillage is known

If: 1) Fallweediness is known, and
2) Fallpredweedtyp is known
Then: 1) Oviposind = the conversion of fallweediness considering fallpredweedtyp to the oviposition attractiveness rating (1.0), and
2) Fallweedknown = yes (1.0)

PREMISE: ($AND (KNOWN CNTXT FALLWEEDINESS)
(KNOWN CNTXT FALLPREDWEEDTYP))
ACTION: (DO-ALL (CONCLUDE CNTXT OVIPOSIND (WEEDTOOVIPOSIND
(VAL1 CNTXT FALLWEEDINESS)
(VAL1 CNTXT FALLPREDWEEDTYP))
TALLY 1000)
(CONCLUDE CNTXT FALLWEEDKNOWN YES TALLY 1000))

RULE022

[This rule is tried in order to find out about the oviposition attractiveness rating for the field in the fall before any tillage]

If: The previous crop of the field is corn
Then: There is strongly suggestive evidence (.8) that the oviposition attractiveness rating for the field in the fall before any tillage is .1

If: Previous_crop = corn
Then: Oviposind = .1 (.8)

PREMISE: ($AND (SAME CNTXT PREVIOUS_CROP CORN))
ACTION: (CONCLUDE CNTXT OVIPOSIND .1 TALLY 800)

RULE023

[This rule is tried in order to find out about the oviposition attractiveness rating for the field in the fall before any tillage]

If: The previous crop of the field is one of: soybeans sweatcorn silage_corn pasture other_than_corn
Then: There is strongly suggestive evidence (.8) that the oviposition attractiveness rating for the field in the fall before any tillage is .2

If: Previous_crop = {soybeans sweatcorn silage_corn pasture other_than_corn}
Then: Oviposind = .2 (.8)

PREMISE: ($AND (SAME CNTXT PREVIOUS_CROP
(ONEOF soybeans sweatcorn silage_corn pasture other_than_corn)))
ACTION: (CONCLUDE CNTXT OVIPOSIND .2 TALLY 800)

RULE024

[This rule is tried in order to find out about the equivalent tillage that was done on the field in the spring considering tillage date compared to moth flight date]

If: 1) The spring tillage that was done on the field is one of: disc chisel field_cultivator moldboard_plow other, and
2) The date of spring tillage is less than the date of the first moth flight
Then: It is definite (1.0) that the equivalent tillage that was done on the field in the spring considering tillage date compared to moth flight date is the spring tillage that was done on the field

If: 1) Spring_till = {disc chisel field_cultivator moldboard_plow other}, and
2) Springtilldate < mfd
Then: Tillage = spring_till (1.0)

PREMISE: ($AND (SAME CNTXT SPRING_TILL
(ONEOF DISC CHISEL FIELD_CULTIVATOR MOLDBOARD_PLOW OTHER))
(LESSP* (VAL1 CNTXT SPRINGTILLDATE)
(VAL1 CNTXT MFD)))
ACTION: (CONCLUDE CNTXT TILLAGE (VAL1 CNTXT SPRING_TILL)
TALLY 1000)

RULE025

[This rule is tried in order to find out about the equivalent tillage that was done on the field in the spring considering tillage date compared to moth flight date]

If: 1) The spring tillage that was done on the field is none, or
2) The date of spring tillage is greater than or equal to the date of the first moth flight
Then: It is definite (1.0) that the equivalent tillage that was done on the field in the spring considering tillage date compared to moth flight date is none

If: Spring_till = none or Springtilldate >= mfd
Then: Tillage = none (1.0)

```
PREMISE: [$AND ($OR (SAME CNTXT SPRING_TILL NONE)
                    (GREATEQ* (VAL1 CNTXT SPRINGTILLDATE)
                              (VAL1 CNTXT MFD]
ACTION: (CONCLUDE CNTXT TILLAGE NONE TALLY 1000)
```

B.2. Parameter Listing

DAMAGE [FIELD-PARMS]
TRANS: (considering the information that you presented, I would say that * could have Black Cutworm damage)
UPDATED-BY: (RULE001)
UPDATED-IN: (RULE007)

FALLOVIPOSIND [FIELD-PARMS]
EXPECT: POSNUMB
TRANS: (the oviposition attractiveness rating of * in the fall after fall tillage)
USED-BY: (Rules 17 16)
CONTAINED-IN: (Rules 17 16)
UPDATED-BY: (Rules 20 19 18)

FALLPREDWEEDTYP [FIELD-PARMS]
EXPECT: (BROADLEAVES GRASSES)
TRANS: (the predominant weed type in * in the fall)
PROMPT: (Which one of the following classes of weeds predominated * in the fall : broadleaves or grasses?)
ASKFIRST: T
USED-BY: (RULE021)

FALLWEEDINESS [FIELD-PARMS]
EXPECT: (COMPLETELY_BARREN NEARLY_VOID NEARLY_VOID-MATURING SPARSE SPOTTY PATCHY PATCHY-EXTENDED WIDESPREAD WIDESPREAD-EXTENDED EXTENSIVE DENSE ELABORATE)
TRANS: (the weediness of * in the fall)
PROMPT: (What was the weediness of * in the fall? You can enter ELABORATE to see elaboration on the categories of weediness.)
ASKFIRST: T
CHECK: [bind TRIAL do
(COND [(EQUAL VALU 'ELABORATE)
(SETQ TRIAL
(GETPROP
(USER.CHOICE
"Type in the number corresponding to the value you wish elaborated."
(REMOVE
'ELABORATE
(GETPROP PARM
'EXPECT))

```
                                NIL NIL PARM)
                        'TRANS))
                (COND ((NULL TRIAL)
                       (RETURN NIL))
                      (T (SPRINTT TRIAL LEFT 5)
                         (RETURN (CAAR (READANS CNTXT PARM
                                               PROMPT]
               (T (RETURN VALU]
    USED-BY: (RULE021)
    CONTAINED-IN: (RULE021)

FALLWEEDKNOWN [FIELD-PARMS]
    TRANS: (the weediness in * in the fall before fall tillage is known)
    USED-BY: (Rules 20 19)
    UPDATED-BY: (RULE021)

FALL_TILL [FIELD-PARMS]
    EXPECT: (NONE DISC CHISEL FIELD_CULTIVATOR MOLDBOARD_PLOW OTHER)
    TRANS: (the fall tillage that was done on *)
    PROMPT: T
    ASKFIRST: T
    USED-BY: (Rules 20 19 18)

FALL_TILL1 [FIELD-PARMS]
    TRANS: (I have considered tillage in the fall)
    USED-BY: (Rules 17 16)
    UPDATED-BY: (RULE018)

FIELD [CONTEXTTYPES]
    PARMGROUP: FIELD-PARMS
    PRINTID: FIELD-
    INITIALDATA: (FIELD-ID WEATHER_STATION PAST-HISTORY)
    GOALS: (DAMAGE)
    DISPLAYRESULTS: T
    SYN: (((FIELD-ID)
          (FIELD-ID)))
    UNIQUE: T
    RULETYPES: (FIELDRULES)

FIELD-ID [FIELD-PARMS]
    ASKFIRST: T
    EXPECT: ANY
    TRANS: (the identifying name of the field under consideration)
    PROMPT: T

FLDFLD [FIELD-PARMS]
    TRANS: (* has been flooded)
    PROMPT: T
    ASKFIRST: T
    USED-BY: (RULE012)
```

```
FLOOD [FIELD-PARMS]
     TRANS: (flooding has occurred to cause favorable conditions for
             sufficient oviposition to cause cutworm damage)
     USED-BY: (RULE002)
     UPDATED-BY: (RULE012)

MFD [FIELD-PARMS]
     EXPECT: ANY
     UNITS: DAYS
     TRANS: (the date of the first moth flight)
     PROMPT: (What is the date of the first moth flight? Enter response
              in the form: (E (SUBSTRING (DATE)
                                         1 9))
              (If you don't know this date, answer NOTKNOWN.))
     ASKFIRST: T
     CHECK: (COND [(EQUAL VALU 'NOTKNOWN)
              [SETQ VALU (CADR (GETPROP (VAL1 CNTXT WEATHER_STATION)
                                        (MKATOM (CONCAT "YR" YR]
              (COND ((EQUAL VALU NIL)
                     (SPRINTT
            "I don't know this date. You will have to provide it."
                             0 5)
                     (CAAR (READANS CNTXT PARM PROMPT)))
                    (T (JDATE VALU]
             (T (CHKDATE VALU)))
     USED-BY: (Rules 25 24 12)

OVIPOS [FIELD-PARMS]
     TRANS: (cutworm eggs have been laid to a sufficient degree to cause
               damage)
     USED-BY: (RULE001)
     UPDATED-BY: (RULE002)

OVIPOSIND [FIELD-PARMS]
     EXPECT: POSNUMB
     RANGE: (0 1)
     TRANS: (the oviposition attractiveness rating for * in the fall
             before any tillage)
     USED-BY: (Rules 20 19)
     CONTAINED-IN: (Rules 20 19)
     UPDATED-BY: (Rules 23 22 21)

PAST-HISTORY [FIELD-PARMS]
     TRANS: (* has had a past history of Black Cutworm damage)
     PROMPT: T
     ASKFIRST: T
     ANTECEDENT-IN: (RULE007)

PLANTDATE [FIELD-PARMS]
     EXPECT: ANY
     UNITS: DAYS
     TRANS: (the estimated planting date)
     PROMPT: (What is the estimated planting date? Enter response in this
              form: (E (SUBSTRING (DATE)
```

1 9)))
ASKFIRST: T
CHECK: (CHKDATE VALU)
USED-BY: (Rules 6 4)
ANTECEDENT-IN: (RULE005)

PREVIOUS_CROP [FIELD-PARMS]
EXPECT: (CORN SOYBEANS SWEATCORN SILAGE_CORN PASTURE OTHER_THAN_CORN)
TRANS: (the previous crop of *)
PROMPT: T
ASKFIRST: T
USED-BY: (Rules 23 22)

RECEDEDATE [FIELD-PARMS]
EXPECT: ANY
UNITS: DAYS
TRANS: (the date that flooding receded)
PROMPT: (What is the date that flooding receded? Enter response in the form: (E (SUBSTRING (DATE)
1 9)))
ASKFIRST: T
CHECK: (CHKDATE VALU)
USED-BY: (RULE012)

SPRINGOVIPOSIND [FIELD-PARMS]
EXPECT: POSNUMB
TRANS: (the oviposition attractiveness rating of * after spring tillage)
USED-BY: (RULE013)
UPDATED-BY: (Rules 17 16 15 14)

SPRINGPREDWEEDTYP [FIELD-PARMS]
EXPECT: (BROADLEAVES GRASSES)
PROMPT: (Which one of the following classes of weeds predominates * in the spring : broadleaves or grasses?)
ASKFIRST: T
USED-BY: (RULE014)

SPRINGTILLDATE [FIELD-PARMS]
EXPECT: ANY
UNITS: DAYS
TRANS: (the date of spring tillage)
PROMPT: (What is the date of spring tillage? Enter response in the form of: (E (SUBSTRING (DATE)
1 9)))
ASKFIRST: T
CHECK: (CHKDATE VALU)
USED-BY: (Rules 25 24 6)

SPRINGWEEDINESS [FIELD-PARMS]
EXPECT: (COMPLETELY_BARREN NEARLY_VOID NEARLY_VOID-MATURING SPARSE SPOTTY PATCHY PATCHY-EXTENDED WIDESPREAD WIDESPREAD-EXTENDED EXTENSIVE DENSE ELABORATE)
TRANS: (the weediness of * in the spring)

```
    PROMPT: (What is the weediness of * in the spring? You can enter
             ELABORATE to see elaboration on the categories of
             weediness.)
    ASKFIRST: T
    CHECK: |bind TRIAL do
            (COND |(EQUAL VALU 'ELABORATE)
                (SETQ TRIAL
                    (GETPROP
                     (USER.CHOICE
  "Type in the number corresponding to the value you wish elaborated."
                                (REMOVE
                                 'ELABORATE
                                 (GETPROP PARM
                                          'EXPECT))
                                NIL NIL PARM)
                         'TRANS))
                (COND ((NULL TRIAL)
                    (RETURN NIL))
                    (T (SPRINTT TRIAL LEFT 5)
                      (RETURN (CAAR (READANS CNTXT PARM
                                             PROMPT|
                (T (RETURN VALU|
    USED-BY: (RULE014)
    CONTAINED-IN: (RULE014)

SPRINGWEEDINESS1 |FIELD-PARMS|
    TRANS: (I have considered spring weediness)
    USED-BY: (RULE013)
    UPDATED-BY: (RULE014)

SPRING_TILL |FIELD-PARMS|
    EXPECT: (NONE DISC CHISEL FIELD_CULTIVATOR MOLDBOARD_PLOW OTHER)
    TRANS: (the spring tillage that was done on *)
    PROMPT: T
    ASKFIRST: T
    USED-BY: (Rules 25 24 6)
    CONTAINED-IN: (RULE024)

SPRING_TILL1 |FIELD-PARMS|
    TRANS: (I have considered spring tillage)
    USED-BY: (RULE013)
    UPDATED-BY: (RULE015)

SURVIVE |FIELD-PARMS|
    TRANS: (cutworm larvae are able to survive on the present weed
            population)
    USED-BY: (RULE001)
    UPDATED-BY: (RULE006)

SYNCHRONY |FIELD-PARMS|
    TRANS: (cutworm development is synchronized with corn development)
    USED-BY: (RULE001)
    UPDATED-BY: (RULE004)
```

TILLAGE [FIELD-PARMS]
EXPECT: (ONEOF NONE DISC CHISEL FIELD_CULTIVATOR OTHER)
TRANS: (the equivalent tillage that was done on * in the spring considering tillage date compared to moth flight date)
USED-BY: (Rules 17 16 15)
UPDATED-BY: (Rules 25 24)

WEATHER_STATION [FIELD-PARS]
EXPECT: (URBANA MT.CARROLL BELLEVILLE)
TRANS: (the closest weather station to *)
PROMPT: (What is the closest weather station to * ? Enter a ? to see the allowed weather stations.)
ASKFIRST: T

WEEDOVIPOSSUFF [FIELD-PARMS]
TRANS: (the oviposition attractiveness rating for the weeds in * is great enough to indicate a Black Cutworm problem)
USED-BY: (RULE002)
UPDATED-BY: (RULE013)

B.3. Pseudo-parameter Listing

COMPLETELY_BARREN [SPECIALS]
TRANS: (COMPLETELY BARREN of all vegetation. Small amounts of residue on the surface.)
ATTRACTIVENESS: (0.0 0.0)

DENSE [SPECIALS]
TRANS: (DENSE cover. Vegetation covers nearly all of the field and most weeds are mature. Weed height may exceed 6%". The vegetation may be functioning as a green manure cover crop.)
ATTRACTIVENESS: (.8 .4)

EXTENSIVE [SPECIALS]
TRANS: (EXTENSIVE vegetation. Vegetation covers most of the field. Most weeds are developing rapidly.)
ATTRACTIVENESS: (1.0 .5)

FDDTOJDATE [FUNCTIONS]
TRANS: (the Julian date that corresponds to (1) FDD that is based on (2))
TEMPLATE: IGNORE
CALLED-IN: (Rules 6 5 4)

FIELDRULES [RULEGROUPS]
CONTEXT: (FIELD)
SVAL: (this field)
CTRANS: "fields"

HEADER [FUNCTIONS]
TRANS: (You have displayed the header : (1))
TEMPLATE: IGNORE
CALLED-IN: (Rules 13 6 4)

NEARLY_VOID [SPECIALS]
TRANS: (NEARLY VOID of vegetation. A few very small
(1/4%" to 1/2%")
weeds can be found with difficulty)
ATTRACTIVENESS: (.1 .05)

NEARLY_VOID-MATURING [SPECIALS]
TRANS: (NEARLY VOID of vegetation. A few small
(1/2%" to 1 1/2%")
weeds can be found with effort. This category
represents a maturing of the category NEARLY_VOID or
possibly may arise from chiseling.)
ATTRACTIVENESS: (.2 .1)

PATCHY [SPECIALS]
TRANS: (PATCHY vegetation. Weeds can be found in small patches and
clumps, some of which are beginning to become lush.
Toward the outer edges of the patches most weeds are
small and recently germinated.)
ATTRACTIVENESS: (.8 .4)

PATCHY-EXTENDED [SPECIALS]
TRANS: (PATCHY vegetation - EXTENDED. This type represents a
maturing of PATCHY. The weeds are growing and
becoming very lush. New germination may or may not
be filling in the area between patches.)
ATTRACTIVENESS: (.9 .45)

SPARSE [SPECIALS]
TRANS: (SPARSE vegetation. A few weeds (larger than 1%")
are scattered throughout the area. This category
represents either a maturing of the category
NEARLY_VOID-MATURING or may result from less than
clean tillage. Chiseling and disking frequently leave
fields in this state.)
ATTRACTIVENESS: (.4 .2)

SPOTTY [SPECIALS]
TRANS: (SPOTTY vegetation. Weeds can be found in small clumps, most
of which will be recently germinated. The clumps are
apparent when walking the field, but may not be
apparent from a distance. This type largely
represents early spring or late fall germination.)
ATTRACTIVENESS: (.6 .3)

WEEDTOOVIPOSIND [FUNCTIONS]
TRANS: (the conversion of (1)
considering
(2)
to the oviposition attractiveness rating)
TEMPLATE: (PARM1 PARM2)
CALLED-IN: (Rules 21 14)

WIDESPREAD [SPECIALS]
TRANS: (WIDESPREAD vegetation. Vegetation covers much of the field,
and the field is beginning to take on a greenish
tinge. Most plants are fairly small
(1/2%" to 1%")
%.)
ATTRACTIVENESS: (.95 .5)

WIDESPREAD-EXTENDED [SPECIALS]
TRANS: (WIDESPREAD vegetation EXTENDED. Vegetation covers much of
the field, and the field takes on a greenish tinge at a
distance. Most plants are lush and growing well.)
ATTRACTIVENESS: (1.0 .5)

B.4. Functions

```
__pp JDATE]

(JDATE
 [LAMBDA (STR) **COMMENT**   **COMMENT**
  (bind STR1 START NOW DIFF
    do (SETQ STR1 (COLLECTPIECES STR))
       (SETQ YR (CADDR STR1))
       (SETQ START (CONCAT "01-JAN-" YR " 00:00:00"))
       (SETQ NOW (CONCAT (CAR STR1)
                         "-"
                         (CADR STR1)
                         "-"
                         (CADDR STR1)
                         " 00:01:00"))
       (SETQ DIFF (FQUOTIENT (IDIFFERENCE (IDATE NOW)
                                          (IDATE START))
                             (ITMES 24 3600)))
       (SETQ DIFF (FQUOTIENT DIFF 3.034074))
       (SETQ YR (MKATOM YR))
       (RETURN (ADD1 (FIX DIFF])
JDATE
```

```
__pp FDDTOJDATE]

(FDDTOJDATE
 [LAMBDA (DD BASEDATE) **COMMENT**   **COMMENT**
  (bind DD1 P1 P2
    do [SETQ P1 (CAR (GETPROP (VAL1 CNTXT WEATHER_STATION)
                      'EQN]
       [SETQ P2 (CADR (GETPROP (VAL1 CNTXT WEATHER_STATION)
                       'EQN]
       (SETQ DD1 (EXPT (FQUOTIENT BASEDATE P1)
                       (FQUOTIENT 1.0 P2)))
       (RETURN (FIX (FTIMES P1 (EXPT (FPLUS DD1 DD)
                                     P2])
FDDTOJDATE

__pp WEEDTOOVIPOSIND]

(WEEDTOOVIPOSIND
 [LAMBDA (WEEDINESS PREDWEEDTYP) **COMMENT**
  (COND
    [(EQUAL PREDWEEDTYP 'BROADLEAVES)
     (CAR (GETPROP WEEDINESS 'ATTRACTIVENESS])
    (T (CADR (GETPROP WEEDINESS 'ATTRACTIVENESS])
WEEDTOOVIPOSIND

__pp CHKDATE]

(CHKDATE
 [LAMBDA (DATE) **COMMENT**
  (bind PARTS TRIAL
    do (SETQ PARTS (COLLECTPIECES DATE))
       (COND
         [(NULL (IDATE (CONCAT DATE " 00:00:00")))
          (COND
            [[AND (EQUAL (LENGTH PARTS)
                         3)
                  (NULL (MEMBER (CADR PARTS)
                                '(MAR APR MAY JUN]
             (SETQ TRIAL
              (FIXSPELL (CADR PARTS)
                        50
                        '(MAY APR MAR JUN)
                        T))
             (COND
              ((NULL TRIAL)
               (RETURN NIL))
              (T [SETQ TRIAL (MKATOM (CONCAT
                                       (CAR PARTS)
                                       "-" TRIAL "-"
                                       (CADDR PARTS]
                 (COND
                  ((NULL (IDATE (CONCAT TRIAL
                                        " 00:00:00")))
                   (RETURN NIL))
```

```
                (T (RETURN (JDATE TRIAL]
          (T (RETURN NIL]
       (T (RETURN (JDATE DATE]))
CHKDATE
```

B.5. Properties of Some Atoms

```
__printprops[URBANA ALL]
EQN : (32.4 .237)
YR80 : (MFD "06-APR-80")
NIL
__printprops[MT.CARROLL ALL]
EQN : (46.2 .192)
YR80 : (MFD "07-APR-80")
NIL
__printprops[BELLEVILLE ALL]
EQN : (16.5 .323)
YR80 : (MFD "28-MAR-80")
NIL
```

REFERENCES

[Baim 1982]
Baim, P.W. "The PROMISE Method for Selecting most Relevant Attributes for Inductive Learning Systems." File No. 898. Urbana: Department of Computer Science, University of Illinois. Sept. 1982.

[Baim 1983]
Baim, P.W. "Automated Acquisition of Decision Rules: The Problems of Attribute Construction and Selection." M.S. Thesis. Urbana: Department of Computer Science, University of Illinois. Sept. 1983.

[Baskin 1980]
Baskin, A.B. "LOGIC NETS: Variable-valued Logic Plus Semantic Networks." *Policy Analysis and Information Systems, No. 3.* 1980.

[Baskin & Michalski 1981]
Baskin, A.B. & R.S. Michalski. "Integrating Multiple-Knowledge Representations and Learning Capabilities in an Expert System: a Study of Theoretical Problems and an Experimental Design." Proposal to Office of Naval Research. Urbana : Department of Computer Science, University of Illinois. March 1981.

[Chilausky 1979]
Chilausky, R.L. "A Prototype Computer Based Consulting System Using the Variable Valued Logic System VL_1: Methodology and Implementation." M.S. Thesis. Urbana: Department of Computer Science, University of Illinois. January 1979.

[Clancey 1977]
Clancey, W.J. "An Antibiotic Therapy Selector which Provides for Explanations." *Proc. of IJCAI5.* 1977.

[Clancey 1981]
Clancey, W.J. "The Epistemology of A Rule-Based Expert System: A Framework for Explanation." Report No. STAN-CS-81-896. Stanford: Stanford University. November 1981.

[Davis & King 1976]
Davis, R., & J. King. "An Overview of Production Systems." *Machine Intelligence 8.* (eds. E.W. Elcock & D. Michie). N.Y.: John Wiley. pp. 300-332. 1976.

[Davis 1981]
Davis, J.H. "CONVART: A Program for Constructive Induction on Time Dependent Data." M.S. Thesis. Urbana: Department of Computer Science, University of Illinois. Sept. 1981.

[Duda et al. 1978]
Duda, R.O., P.E. Hart, N.J. Nilsson, & G.L. Sutherland. *Pattern Directed Inference Systems* (eds. D.A. Waterman & F. Hayes-Roth). N.Y.: Academic Press. 1978.

[Gevarter 1982]
Gevarter, W. "An Overview of Expert Systems." Report No. NBSIR 82-2505. Washington D.C.: National Bureau of Standards. 1982.

[Hart 1982]
Hart, P. E. "Directions for AI in the Eighties." *Sigart*. pp. 11-16. #79. Jan. 1982.

[Kimpel 1978]
Kimpel, H.K. "SIMBLAC - A GASP-IV Model of the Black Cutworm Life Cycle." M.S. Thesis. Lafayette: Department of Industrial Engineering, Purdue University. December 1978.

[Larson 1977]
Larson, J.B. "INDUCE-1: An Interactive Inductive Inference Program in VL_{21} Logic System." Report No. 876. Urbana: Department of Computer Science, University of Illinois. May 1977.

[Michaelsen 1982]
Michaelsen, R. "A Knowledge Based System for Individual Income and Transfer Tax Planning." Ph.D. Thesis. Urbana: Department of Accountancy, University of Illinois. May 1982.

[Michalski 1973]
Michalski, R.S. "AQVAL/1 - Computer Implementation of a Variable-Valued Logic System VL_1 and Examples of its Application to Pattern Recognition." *Proceedings of the First International Joint Conference on Pattern Recognition.* Washington, DC, October 30 - November 1. pp. 3-17. 1973.

[Michalski & Larson 1978]
Michalski, R.S., & J. B. Larson. "Selection of Most Representative Training Examples and Incremental Generation of VL_1 Hypotheses: The Underlying Methodology and the Description of Programs ESEL and AQ11." Report No. 867. Urbana: Department of Computer Science, University of Illinois. May 1978.

[Michalski 1980a]
Michalski, R.S. "Pattern Recognition as Rule-Guided Inductive Inference." *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 4.* pp. 349-361. July 1980.

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

[Michalski et al. 1982]
Michalski, R.S., J.H. Davis, V.S. Bisht & J.B. Sinclair. "Plant/ds: An Expert System for the Diagnosis of Soybean Diseases." In *Proceedings of the 1982 European Conference on Artificial Intelligence.* Orsay France, 12-14 July. pp. 133-138. 1982.

[Michalski et al. 1983a]
Michalski, R.S., A.B. Baskin & A.G. Boulanger. "An Overview of the ADVISE Meta Expert System." To be published.

[Michalski et al. 1983b]
Michalski, R.S., A.B. Baskin, A. Boulanger, R. Reinke, L. Rodewald, M. Seyler, K. Spackman & C. Uhrik. "A Technical Description of the ADVISE Meta Expert System." Report to be published. Urbana: Department of Computer Science, University of Illinois.

[Michalski et al. 1983c]
Michalski, R.S., J.H. Davis, V.S. Bisht & J.B. Sinclair. "A Computer-Based Advisory System for Diagnosing Soybean Diseases in Illinois." Accepted for publication in *Plant Diseases*. 24pp.

[Michie 1980]
Michie, D. "Knowledge-based Systems" Report No. 1001. Urbana: Department of Computer Science, University of Illinois. January 1980.

[Nilsson 1980]
Nilsson, N. *Principles of Artificial Intelligence*. Palo Alto: Tioga Publishing Co. 1980.

[Schubert 1977]
Schubert, R. "The VL Relational Data Sublanguage for an Inferential Computer Consultant." M.S. Thesis and Report No. 846. Urbana: Department of Computer Science, University of Illinois. October 1977.

[Sherrod et al. 1979]
Sherrod, D.W., J.T. Shaw, & W.H. Luckmann. "Concepts on Black Cutworm Field Biology in Illinois." *Environmental Entomology, Vol 8*. pp. 191-195. 1979.

[Shortliffe 1976]
Shortliffe, E.H. *Computer-Based Medical Consultations: MYCIN*, N.Y.: Elsevier. 1976.

[Spackman 1983]
Spackman, K. "QUIN: Integration of Inferential Operators in a Relational Data Base." M.S. Thesis. Urbana: Department of Computer Science, University of Illinois. May 1983.

[Stefik et al. 1982]
Stefik, M., J. Aikins, R. Balzer, J. Benoit, L. Birnbaum, F. Hayes-Roth, & E. Sacerdoti. "The Organization of Expert Systems: A Prescriptive Tutorial. Report No. VLSI-82-1. Palo Alto: Palo Alto Research Centers, Xerox Corp. January 1982.

[Stepp 1980]
Stepp, R., "Learning from Observation: Experiments in Conceptual Clustering." *Proceedings Machine Learning Workshop-Symposium*. Carnegie-Mellon University, Computer Science Department. July 16-18, 1980.

[Troester 1982a]
Troester, S.J. "Damage and Yield Reduction in Field Corn due to Black Cutworm Feeding: Results of a Computer Simulation Study." *J. Econ. Entomol.* Vol. 75, pp. 1125-1131. 1982.

[Troester et al. 1982b]
Troester, S.J., S.L. Clement, W.B. Showers, & A.J. Keaster. "Determining Yield Loss by Black Cutworms on Corn." *ASAE paper No. 82-5026*. St Joseph, Mi. 1982.

[Troester et al. 1982c]
Troester, S.J., W.G. Ruesink, & R.W. Rings. "A Model of Black Cutworm *(Agrotis ipsilon)* Development: Description, Uses, and Implications." *Ill. Agric. Exper. Stn. Bull., No. 774*. 33pp. 1982.

[Troester et al. 1982d]
Troester, S.J., C.D. Bremer, K.L. Steffey, D.E. Kuhlman, & R.H. Meyer. "Avoiding Black Cutworm Losses: An Educational Manual for a Computer Model that Demonstrates Effects of Corn Production Practices on Black Cutworms." Mimeograph. Champaign: Sect Econ. Entomol., Illinois Natural History Survey. 30pp. 1982.

[Troester et al. 1983]
Troester, S.J., R.H. Meyer, K.L. Steffey, G.L. Godfrey, C.B. Bremer, S.P. Briggs, D.E. Kuhlman, & J.T. Shaw. "Projecting the Beginning of Black Cutworm Damage." *J. Econ. Entomol.* Vol. 76. pp. 362-367. 1983.

[van Melle 1979]
van Melle, W. "A Domain-Independent Production-Rule System for Consultation Programs." Report No. HPP-79-7. Stanford: Department of Computer Science, Stanford University. 1979.

[van Melle et al. 1981]
van Melle, W., A.C. Scott, J.S. Bennett, & M. Peairs. "The EMYCIN Manual." Report No. STAN-CS-81-885. Stanford: Department of Computer Science, Stanford University. October 1981.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-83-1134	2.	3. Recipient's Accession No.
4. Title and Subtitle: THE EXPERT SYSTEM PLANT/CD: A CASE STUDY IN APPLYING THE GENERAL PURPOSE INFERENCE SYSTEM ADVISE TO PREDICTING BLACK CUTWORM DAMAGE IN CORN			5. Report Date: July 1983 6.
7. Author(s): Albert Gerard Boulanger			8. Performing Organization Rept No.
9. Performing Organization Name and Address: Department of Computer Science University of Illinois Urbana, IL 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No. N00014-82-K-0186
12. Sponsoring Organization Name and Address: Office of Naval Research Arlington, VA			13. Type of Report & Period Covered 14.

15. Supplementary Notes

16. Abstracts

This report describes the expert system PLANT/cd, which is a case study in applying the general purpose inference system ADVISE to predicting Black Cutworm damage in corn. This application addresses the interesting problem of linking surface and deep models of a domain. Included in this report is a general description of the general inference tool, ADVISE, a general description of the Black Cutworm damage domain, a description of the control scheme, rules, and simulation functions used to implement PLANT/cd, and some verifying experiments of the system. Also described is an implementation in EMYCIN, PLANT/cdp, that is used to suggest whether a field could have Black Cutworm damage.

17. Key Words and Document Analysis. 17a. Descriptors

Rule-Based Expert Systems, Expert Systems, AI Programs, Pest Management, Pest Simulation, Pest Modeling, Corn Pests, Black Cutworm, Black Cutworm Damage in Corn, Knowledge Representation, Surface and Deep Models

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 127
	20. Security Class (This Page) UNCLASSIFIED	22. Price