1-1-1983

An overview of expert systems.

Jane Greenberg Cohen

AN OVERVIEW OF EXPERT SYSTEMS

by

JANE GREENBERG COHEN

A Thesis

Presented to the Graduate Committee

in Candidacy for the Degree of

Master of Science

in

Computing Science

Division of Computing and Information Science

September, 1983

Lehigh University

Certificate of Approval

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

Sept. 9 1983
(date)

Professor in Charge

Chairman of Department

Acknowledgments

I wish to thank my husband for his continual support and Professor Rayna for his professional guidance in preparing this thesis.

Table of Contents

List of Figures

Abstract

This paper is an overview of expert systems. Expert systems are Artificial Intelligence programs embodying the specific knowledge and experience of human experts. They offer users advice in specialized domains that are generally conceded to be difficult and requiring expertise.

The components of an expert system are the knowledge base, a knowledge representation subsystem, a knowledge acquisition subsystem, an inference driver, an explanation capability and a natural language front end translator.

The organization of an expert system is determined by the solution space available, errors in the data, and the availability of abstraction in the system.

In this paper the author describes software tools, an area of expert systems that is receiving increasing attention. Special purpose programming languages, such as, LISP and PROLOG allow knowledge bases to be built more efficiently. EXPERT and EMYCIN are software programs aiding in the development of production systems. Another software tool, SAUI provides the user with help in learning and using complex expert systems. SAUI is a software package for general development for expert systems. Software tools available are becoming more and

more sophisticated.

Some recent areas providing practical applications for expert systems are medical diagnosis, equipment failure diagnosis, computer configuration, chemical data interpretation and structure, oil field service, and military needs.

Areas of research in the study of expert systems that are described in this thesis are heuristics, knowledge acquisition and knowledge representation. Heuristics are informal judgmental 'rules of thumb'. Knowledge representation refers to the method of representing an ordered set of task specific rules in an expert system. Acquiring new knowledge or refining existing knowledge is accomplished by the knowledge acquisition methods of an expert system.

This paper includes a comparison of two expert systems. CENTAUR and R1 have many differences and few similarities in their methods of representing, acquiring, utilizing, and explaining knowledge. R1 is an expert system used to configure VAX-11/780 computer systems. CENTAUR interprets pulmonary tests and diagnosis pulmonary diseases. The capabilities of CENTAUR and R1 demonstrate the potential of expert systems in the future.

1. Introduction

Expert systems are computer programs that embody the specific knowledge of human experts. They are one of the most practical products to have come out of research in Artificial Intelligence. The expert system provides useful answers to questions asked by the user in a field, such as, medical diagnosis.

This thesis attempts to provide an overview of expert systems. This first chapter is an introduction. The second chapter consists of a description of expert systems. It contains the definition, history, components, organization, software tools, and applications of expert systems.

There are several major research issues in studying expert systems. They include heuristics, knowledge acquisition and knowledge representation. These issues are discussed and illustrated in Chapter three.

The final chapter compares two expert systems. They are R1 and CENTAUR. R1 is a configuration system for the VAX-11/780 computer systems. When provided with a customer's order, it produces as output diagrams configuring the components on the order. CENTAUR is an expert system that interprets measurements from pulmonary function tests administered to patients in a pulmonary function labratory. It then produces a set of interpretation statements and a diagnosis for each patient. CENTAUR and R1 have more differneces that are discussed then similarities.

Expert systems are a human face of information technology and will find an application in every sector and level of modern economy.

2. Description of Expert Systems

The beginning section of this chapter defines expert systems. The intermediate sections include the history, components, organization, and applications of expert systems. The final section describes software tools used in expert systems.

2.1 Definition of Expert Systems

A most powerful technique for exploiting collective human knowledge by computer is represented by what are called Expert Systems [Pinkerton, 1982]. Expert systems are Artificial Intelligence computer programs that embody the specific knowledge and experience of human experts. They are problem-solving programs that solve substantial problems generally conceded as being difficult and requiring expertise [Stefik et al, 1982] for which 'good' algorithms are not known. [Chester, 1982] Their goal is to provide users with advice in specialized domains [Bonnet, Cordier, and Kayser, 1981].

Expert systems encapsulate the knowledge of one or more experts in a particular field. These systems consist of a global data base of assertions, a set of rules that represent small bits of an expert's knowledge, a control strategy for applying the rules to the assertions [Chester, 1982], a knowledge acquisition program, an explanation program, and a natural language processor [Winfield, 1982].

The system may gradually improve its performance as it is used, provided its users are true experts. It does this by adding to the data base and also be refining the 'rules' by which the system works. These rules are listed through dialogues with experts who may be able to point out, from their own special knowledge of cases, the flaws in the generalizations represented by the rules. own special knowledge of cases. Thus an expert system can evolve to a degree where its general performance is as good as that of a group of experts collectively, possibly faster than any of them, and certainly better than any inexperienced user. In this way the experts' knowledge and judgment is indirectly made much more widely available [Pinkerton, 1982].

The expert system performs its inferences using a human-like process, and must be capable of explaining its inference processes in a language natural to the user, if it is to be acceptable to the user. If an expert system is to follow a similar problem-solving process as a human, and yet it is to run on a computer it was suggested by Basden at a recent conference on expert systems, that this relationship be represented as follows:

Expert

Numerical computer	system	Human [capable of processing structured information]

The aim of research into expert sytems is to move much closer to the human end of the spectrum [Winfield, 1982].

The process of designing and implementing expert systems is known as Knowledge Engineering [Tanaka, 1982]. Feigenbaum defines the activity of knowledge engineering as follows:

> "The knowledge engineer practices the act of bringing the principles and tools of AI research to bear on difficult applications problems requiring experts' knowledge for their solution. The technical issue of acquiring this knowledge, representing it, and using it appropriately to construct and explain lines of reasoning, are important problems in the design of knowledge based systems....The art of constructing intelligent agents is part of an extension of the programming art. It is the art of building complex computer programs that represent and reason with the knowledge of the world. [Feigenbaum, 1977]

The basic idea of expert systems is putting knowledge to work, a non-mathematical knowledge used for most of the world's problems. The knowledge base of an expert system includes a data base consisting of facts, assumptions and beliefs, and heuristic rules [Feigenbaum, 1982].

The heuristic approach rather than an algorithmic approach characterizes an expert system. The system searches for a good enough answer with the resources available using the knowledge of a human expert to improve search efficiency. This permits investigation of feasible modes only and the rapid elimination of "blind alleys". [Sumner, 1982]

For expert systems, logic is not the issue, knowledge is. These systems, of course, needs an inference procedure; however, the power of an expert system comes from its knowledge, not its inference procedure.

2.2 History of Expert Systems

Twenty years ago Newell [Newell, 1962] surveyed several organizational alternatives for problem solvers. He was concerned with how one should go about designing problem solving systems [Stefik et al, 1982]. The research that followed in the area of computer problem solving passed through various stages. In the first phase attempts were made to improve on human problem solving performance by using various statistical techniques [Couch, 1976]. Statistical methods proved to be accurate for small diagnostic domains, but impractical for applications in real world problems [Gorry, 1976]. In the second phase attempts were made to capture diagnostic logic as fixed decision protocols using an inference-based paradigm [Bleich, 1969]. Although at times successful, it was recognized that such protocols suffered from inflexibility. Along the way, Artificial Intelligence researchers made an important discovery. The power of an intelligent program to perform its task depends primarily on the quantity and quality of knowledge it has about that task [Davis and Lenat, 1982].

This observation arises not only in the work of artifact builders but in the work of psychologists, for example, the studies of Simon and his colleagues on the nature of "expert" thought in physics and chess playing; in the work of the image understanding researchers; and the work on understanding natural language. Human specialists striving for high levels of proficiency in their chosen fields spend years acquiring the knowledge and skills necessary to support such performance.

Thus knowledge came to be seen of paramount importance, and Artificial Intelligence research shifted its focus from an inference-based paradigm to a knowledge-based paradigm. Knowledge is viewed as consisting of facts and heuristics. The facts constitute a body of information that is widely shared, publicly available, and generally agreed upon by experts in the field. The heuristics are more private, little discussed rules of good judgement. They are rules of good guessing and plausable reasoning that characterize expert level decision making in a field.

Beginning in 1965, the Stanford Heuristic Programming Project focused on the development and exploitation of the knowledge based paradigm. It began in artifact construction and methodological innovation with the DENDRAL program, with efforts directed towards building a system which incorporated expert problem solving strategies, but which retained flexibility. DENDRAL solved problems of structure elucidation in organic chemistry, initially by a knowledge intensive analysis of physical spectra of the molecules [Davis and Lenat, 1982].

In 1968 when Feigenbaum presented the research work on DENDRAL [Lindsay et al, 1980] to Michie, Professor at Edinburgh University in Scotland, the term of Epistemological Engineering was proposed by Michie to describe such research works as DENDRAL [Tanaka, 1982]. Epistemics is the science of communicating understanding via stored knowledge [Addis, 1982].

The descendant of DENDRAL, META-DENDRAL, analyzed sets of spectral data and inferred chemical rules of spectral analysis. It created knowledge from data, guided by a few basic principles.

The first foray into clinical medicine was the MYCIN effort. MYCIN was a program that performed consultations with physicians about infectious disease diagnosis and antimicrobial therapy. The production rule representation that proved so effective in parts of DENDRAL was adapted to fit the needs of medical knowledge. Issues in machine-facilitated knowledge acquisition, in the representation of knowledge, and in program control arose from this work [Davis and Lenat, 1982]. MYCIN provided an inspiration for Davis' work on TEIRESIAS, a program designed to make possible the interactive transfer of expertise from a human expert to the knowledge base of a high performance program, in a dialogue conducted in a restricted subset of natural language [Davis and Lenat, 1982].

Another approach to knowledge representation was initiated by Minsky's theory of frames, explained by Minsky in 1974. A frame is a data-structure for representing a stereotyped situation [Minsky, 1974]. Minsky's frames were incorporated in the development of systems, such as, AM - a program that models an aspect of elementary mathematics research in the development of new concepts under the guidance of a body of heuristic rules [Lenat, 1982]; and CENTAUR- a program designed to diagnose pulmonary disease [Aikins, 1983].

In August, 1977, Professor Feigenbaum presented a paper titled "The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering" at the 5th International Joint Conference on Artificial Intelligence held at MIT. Because of the difficulty of pronouncing Episemological Engineering, Knowledge Engineering was selected at this time, to describe the process of designing and implementing expert systems [Tanaka, 1982].

Since the inception of Expert Systems many have been written. Some expert systems that have come into regular use are DENDRAL [Lindsay et al, 1980]; MACSYMA [Martin and Fateman, 1971]; PUFF [1980"HeuristicProgrammingProject1980", 1980] and R1 [McDermott, 1980]. Macsyma manipulates algebraic expressions symbolically, including their integration and differentiation; Puff diagnoses pulmonary disorders; and R1 configures VAX systems [Chester, 1982].

Major areas of research in the field of knowledge engineering are the following.

1. Knowledge Base.
2. Knowledge Representation.
3. Knowledge Acquisition.
4. Knowledge Utilization.
5. Knowledge Explanation Subsystem.

2.3 Components of an Expert System

An expert system (figure below) consists of a number of essential components: a knowledge base, a knowledge representation subsystem, a knowledge acquisition subsystem, an inference driver, an explanation capability, and a natural language front end translator.

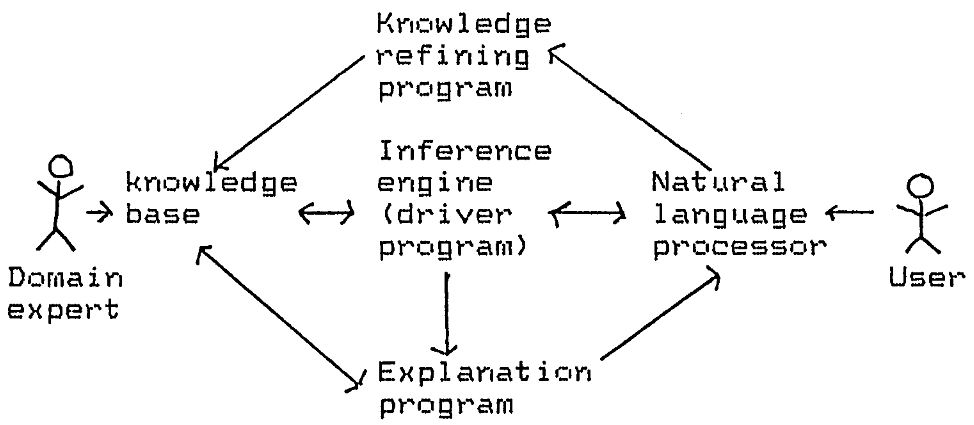

Knowledge refining program
Inference engine (driver program)
knowledge base
Natural language processor
Domain expert
User
Explanation program

Figure 2-1: Components of an Expert System

An expert system is a set of computer programs that access a knowledge base and perform inferences on the knowledge held there, in order to satisfy a user query. The expert system must be capable of explaining its inference processes in a language natural to the user [Winfield, 1982].

The knowledge base is a database of information consisting of facts, assumptions and beliefs, and hueristics that describe the problem to be solved and all the intermediate results in its solution [Chester, 1982]. Generally a knowledge base is composed of information collected in discussions between a human expert and a system builder (also human at present).

A knowledge representation is a set of rules providing a formalism in a data structure, for facts and heuristics about a subject or specialty [Tanaka, 1982]. Usually, an expert system will contain 40 - 800 rules.

Several methods of representing knowledge within a computer are currently used: 1)Logic; 2) Procedural representations; 3)Semantic networks; 4)Production systems; 5)Direct (analogical) representations; 6)Semantic primitives; 7) Frames.

The method which has been used in the majority of the more common present day expert systems is the production system. A production system consists of a number of rules, each rule being of the IF...THEN... type (see figure 2.2). These rules are sometimes referred to as situation action rules. That is IF some situation occurs THEN some action is performed [Winfield, 1982].

```
   IF:          Request is PUTON object targer
        AND     Object is free
        AND     Target is free

 THEN:          Delete Object is free
        AND     Delete Target is free
        AND     Delete PUTON object target
        AND     Move Object to Target
```

Figure 2-2: If...Then... rule

In the modern world it is necessary for humans to update their knowledge by deleting old outdated information, inserting new information, and amending existing information. Similarly, the expert system via its knowledge refining component needs to have its knowledge base updated to ensure that it remains an expert in its field.

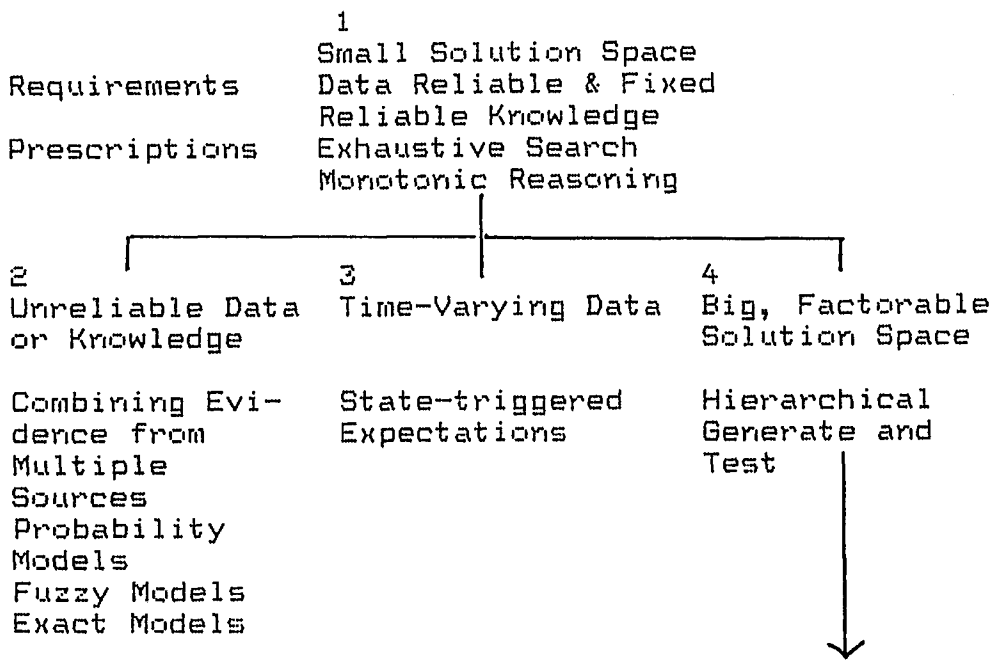
The inference engine is the program of control strategy that drives the system. It provides a methodology for reasoning about rules in a knowledge representation and drawing conclusions from that knowledge [Tanaka, 1982]. It does this by attempting to match known facts about a particular problem with one (or perhaps more) of the productions [Winfield, 1982]. Rules are applied mainly in the backward direction, but sometimes the forward direction. Some systems apply them in both directions [Chester, 1982]. When a successful match is found, the production 'fires' and the action part of the rule is used to update the 'known facts' of the data base. It is unlikely to be able to solve the problem in one step and will therefore attempt to produce a solution to a small part of the problem by setting up a subgoal to be solved. Subgoals are established by writing appropriate notes about them into the data base. Using this new knowledge in conjunction with what was already known about the problem, the knowledge utilization program again attempts to satisfy the goal by finding another production that is satisfied. This process is repeated until a solution is found [Winfield, 1982].

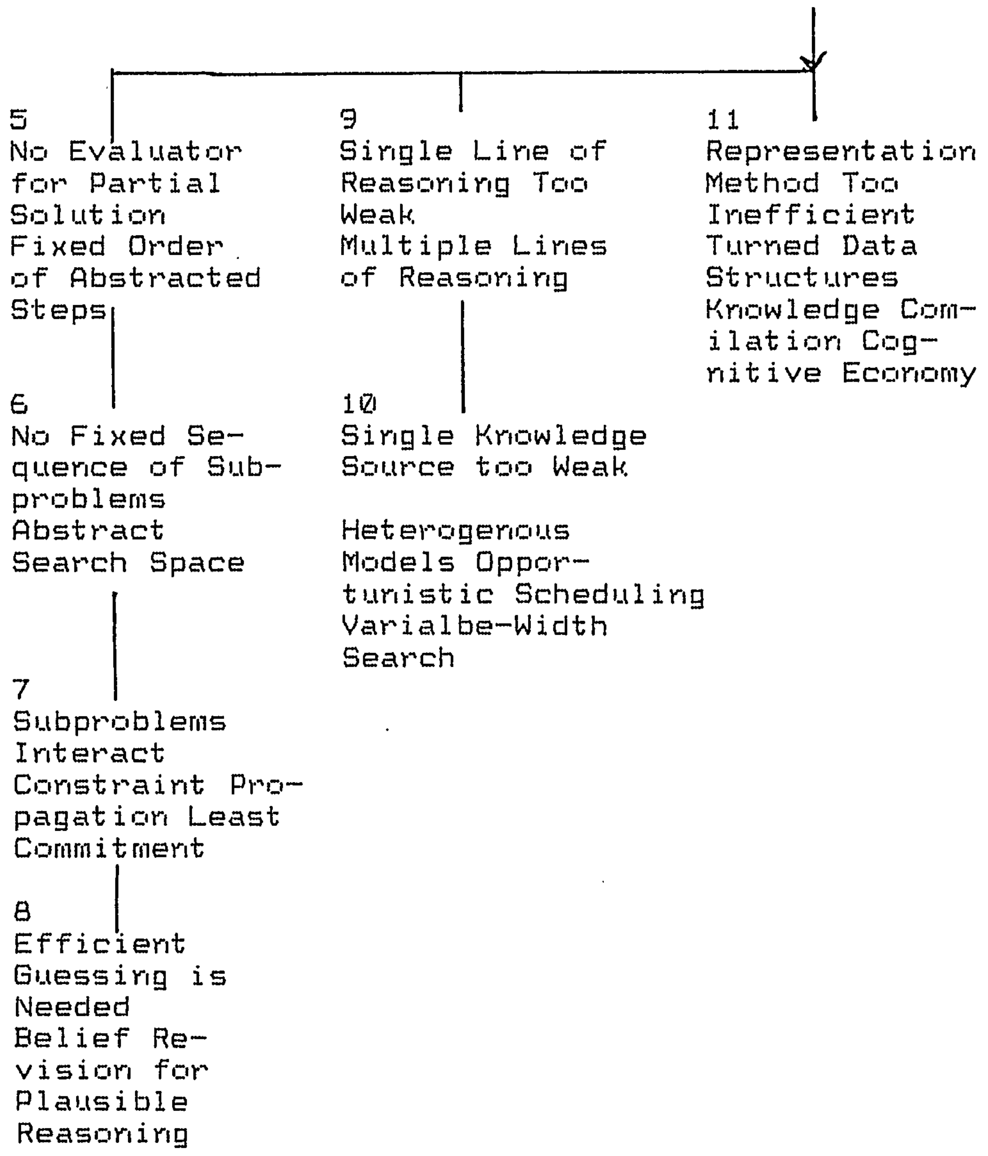
A natural language processor provides acceptable communication between the expert system and the user. Communication must be in natural language that is understandable to the user. The system must make it easy for the user to input requests, and obtain results. The system should also be capable of adjusting the type of questions it asks and the amount and type of information it gives or requests, depending upon whether an expert or naive user is controlling the system.

The natural language front end is the part of the expert system the user comes into contact with and is therefore very important. A poor natural language front end could make the sytsem unacceptable, particularly to naive users, and these are the type of users expert systems will be built for in the future [Winfield, 1982].

2.4 Organization of Expert Systems

In this section a number of contemporary systems are used to illustrate the strengths and limitations of alternative organizational methods of building an expert system. In an expert system the choice of search method is one of the most important decisions. The approach of searching for a solution is affected by characteristics of the domain, such as size of the solution space, errors in the data, and the availability of abstractions.




5 No Evaluator for Partial Solution Fixed Order of Abstracted Steps
6 No Fixed Sequence of Subproblems Abstract Search Space
7 Subproblems Interact Constraint Propagation Least Commitment
8 Efficient Guessing is Needed Belief Revision for Plausible Reasoning
9 Single Line of Reasoning Too Weak Multiple Lines of Reasoning
10 Single Knowledge Source too Weak
Heterogenous Models Opportunistic Scheduling Varialbe-Width Search
11 Representation Method Too Inefficient Turned Data Structures Knowledge Comilation Cognitive Economy

Stefik et al suggested the chart above for the alternative cases of organizing expert systems. Each box in the figure corresponds to one of the cases. The numbers on top of the boxes indicate the order in which the cases are discussed. The cases are organized into a tree structure such that a sequence of cases in a branch refers to increasing elaborate considerations of a basic idea. Case 1 requirements are small solution space, reliable data and exhaustive search. The boxes 2 through 4 consider the complications of unreliable data or knowledge time-varying data, and a large search space. Organizing a given expert system may require combining ideas from any of these topics. The three branches descending from case 4 consider further the problem of a large search space. The first branch (cases 5 through 8) are organizations for abstracting a search space. The third branch considers ways of making the knowledge base more efficient. The organization of the cases is pedagogical and it should be realized that in real system the ideas of the varying branches may be combined.

Case 1- Small Search Space with Reliable Knowledge and Data

This case considers the simplest architecture for an expert system. The first requirement is that the data and knowledge are reliable and not filled with errors. Ths second requirement is that search space is small and provisions are therefore unnecessary to cope with the limitations of computational resources. It should be realized that in real applications few expert systems meet these criteria.

An expert system of this type could be organized into two main parts: a memory and an inference method. The memory would consist of a list of inferred facts that possibly could be represented in predicate calculus [Barr and Feigenbaum, 1980], for example

```
          [On Block1 Block2]
          [NOT [on Block2 Table1]]
```

Figure 2-3: Representation in Predicate Calculus

The data could be stored in a frame system [Bobrow, 1975] where the indexing of facts is organized to make the most common paths more efficient. Data which are used together are stored in the same frame.

Case 2- Unreliable Data or Knowledge

Sometimes it is necessary for expert systems to make a judgement under pressure of time. Some of the knowledge or data can be unreliable or unavailable.

MYCIN is an example of an expert system that approaches reasoning with uncertainty. To accomodate judgmental reasoning MYCIN incorporates concepts such as "A suggests B" or "C and D tend to rule out E" by using numbers called certainty factors to indicate the strength of a heuristic rule. An example of a rule represented in

```
IF (1)the infection is primary-bacteremia and
   (2) the site of the culture is one of the sterile
       sites and
   (3) the suspected portal of entry of the organism
       is the gastro-intestinal tract,
THEN there is suggestive evidence (.7) that the
   identity of the organism is bacteroides.
```

Figure 2-4: MYCIN Rule.

the MYCIN knowledge base is: The number "0.7" indicates the strength of the probability that the hypothesis is true. Evidence for and against the hypothesis is processed separately, and the "truth" of the hypothesis is the algebraic sum of the evidence. [Rundle, 1982]

Instead of using its own formalism for reasoning with uncertainty, MYCIN could have used Bayes' Theorem [Catanzarite, Greenburg and Bremermann, 1981]. It could calculate probability in light of specified evidence, from the a priori probability of the disease and the conditional probabilities relating the observations to the disease. The main difficulty with Bayes Rules is the large amount of data that are required to determine the conditional probabilities needed in the formula.

Another approach to inexact reasoning is fuzzy logic as discussed by Zadeh [Zadeh, 1979] and others. In fuzzy logic, the statement "X is a large number' is interpreted as having an imprecise denotation characterized by a fuzzy set. A fuzzy set is a set of values with corresponding characteristic functions.

```
Fuzzy Proposition:
X is a large number.
Corresponding fuzzy set:
[X is a number [0,10], .1]
[X is a number [10,1000, .2]
[{X > 1000}, .7]
```

The interpretation of the proposition 'X is large' is that if X is less than 10 it has a characteristic function of .1, or between 10 and 1000 a characteristic function of 0.2 and so on.

The usefulness of fuzzy logic in reasoning about unreliable data would depend on the appropriateness of interpreting the data as a fuzzy proposition.

Besides the use of pseudo-probability and fuzzy approaches for reasoning with partial and unreliable data, one could use an exact inference method. This approach is illustrated in the expert system GAI [Stefik, 1978] which is a data interpretation system that copes with errorful data. GAI's task is to assemble models of complete DNA structures using incomplete information about the digestion of molecules by enzymes [Rundle, 1982].

An example of a rule for correcting missing data is:

If a segment appears in a complete digestion for an
   enzyme, that fails to appear in the incomplete
   digestion for that enzyme,
then it may be added to the list of segments for the
   incomplete digestion.

This rule is based on the observation that segments are easier to overlook in incomplete digestions than in complete digestions.

In summary, there are several methods for reasoning with unreliable data and knowledge. All of the methods require a formalization of extra meta-knowledge in order to correct the data, take back assumptions, or combine evidence. The available meta-knowledge is a critical factor in viability of these approaches to a particular application.

Case 3: Time varying data

Some expert tasks involve reasoning about situations that change over time. The change of the situation can be signalled by time varying- data as in the expert system VM (Ventilator Manager) reported by Fagan [Fagan et al, 1979, Fagan, 1980]. VM is a program that interprets the clinical significance of patient data from a physiological monitoring system by monitoring the post-surgical progress of a patient requiring mechanical breathing assistance.

Because a patient's situation can be affected by the progression of disease and the response to theraputic intervention, VM is an application containing knowledge suitable for coping with time-varying data. VM has several kinds of rules: transition rules, initiation rules, status rules and therapy rules. The rules are rerun periodically when VM receives a new set of instrument measures. The following is an example of a transition rule used to detect when a patient's state has changed:

```
If (1) the current context is 'Assist' and
   (2) respiration rate has been stable for 20
       minutes and
   (3) I E ratio has been stable for 20 minutes
Then the patient is on 'CMV' (controlled mandatory
   ventilation)
```

This rule governs a transition between an 'assist' context and a 'CMV' context or state. VM uses initialization rules to update information for a new context and extablish new expectations for status rules.

VM's reasoning is concerned only with the previous state and the next state. It is limited to adjacent time intervals. Research in writing programs capable of reasoning about distant events (requiring elaborate representations of events and time), for example, planning and prediction tasks, is in progress.

Case 4: Large but factorable solution space

This section describes a technique for coping with very large spaces. This technique is necessary when it is not enough to find one interpretation of data, but every interpretation consistent with the data is required.

A systematic approach would be to consider all possible cases and eliminate those cases inconsistent with the data. However, the techniques is impractical. A practical alternative is to use early pruning while generating and testing solutions. Two expert systems using this technique are DENDRAL [Stefik, 1981] and GAI [Stefik, 1978]. GAI was mentioned previously.

DENDRAL generates possible molecular structures from mass spectrometer data, nuclear magnetic renosance data and other information. It works in three stages, using a "generate and test" approach. First it derives a number of constraints which the structure must satisfy, and then generates a number of structures which satisfy these constraints. The proposed structures are then processed to predict their mass spectrogram and these are compared with the observed experimental data. The program has been accurate and used to establish new molecular structures. [Rundle, 1982]

In conclusion, generate and test is appropriate method to consider when it is important to find all solutions to a problem. However, to be workable, the generator must partition the solution space in ways to allow early pruning. Often these criteria are associated with data interpretation and diagnostic expert system problems.

Case 5: No evaluator for partial solutions

In design and planning problems one cannot tell from a fragment of a plan or design whether that fragment is part of a complete solution. This section considers an approach to problem solving without early pruning. The approach uses the technique of abstracting the search space by emphasizing the important steps of a problem in a fixed order. This enable the problem to be partitioned into subproblems.

An illustration of this is the R1 program reported by McDermott [McDermott, 1982]. R1 configures Digital Equipment Corporation's VAX computer system. The input is a customer's order and the output is a set of diagrams displaying the spatial relationship among components on the order. R1 is capable of determining whether a customer's order is satisfactory and adding necessary components if it is not.

The configuration task can be grouped into the following subtasks that have strong temporal interdependance.

1. Determines whether there is anything grossly wrong with the customer's purchase order.

2. Put the appropriate components in the CPU and CPU expansion cabinets.

3. Put boxes in the unibus expansion cabinet and put the appropriate components in those boxes.

4. Put panels in the unibus expansion cabinets.

5. Layout the system on the floor.

6. Do the cabling.

An example of a rule for the third subtask follows:

```
If the most current active context is assigning a power
       supply
   and a unibus adaptor has been put in a cabinet
   and the position it occupies in the cabinet is known
   and there is space available in the cabinet for a
       power supply for that position
   and there is an available power supply
   and there is no H7101 regulator available
Then add an H7101 regulator to the order.
```

Because of the way in which the stages in the process have been abstracted, R1 always processes the tasks in the same order and never needs to backtrack. [Rundle, 1982]. R1, with the use of abstraction space does very little search. This method requires a partial ordering on decisions for a task since the consequences of applying an operator will affect 'later' parts of the solution.

The use of abstractions should be considered for applications where there is a large search space but no method for early pruning.

Case 6: No fixed partitioning of subproblems

This section describes an organization appropriate to applications that cannot with each use be partitioned into the same subproblems. In this type of system an abstract approach is used. The following aspects of this approach are important:

1. Abstractions for each problem are composed from terms (selected from a space of terms) to fit the structure of the problem.

2. During the problem-solving process, these concepts represent partial solutions that are combined and evaluated.

3. The concepts are assigned fixed and predetermined abstraction levels.

4. The problem solution proceeds topdown, that is from the most abstract to the most specific.

5. Solutions to the problems are completed at one level before moving down to the next more specific level.

6. Within each level, subproblems are solved in a problem independent order.

ABSTRIPS [Sacerdoti, 1974] is an example of this approach. The robot planning expert system makes plans for a robot to move objects between rooms. In ABSTRIPS the abstractions are plans. The concepts are type and color, inroom, etc.; and the abstraction levels are represented by what Sacerdoti refers to as criticality values. These values place the concepts in heirarchy of importance. In one example Sacerdoti suggested the following criticality assignents for concepts in a robot planning domain:

Type and Color	4
InRoom	3
Plugged and Unplugged	2
NextTo	1

One should note that in all problems of the the domain, the hierarchy of 'Type and Color' will always be greater than 'InRoom'. Planning in ABSTRIPS starts where criticality is at a maximum. Preconditions whose criticality is below the current level are invisible to the planner and will be accounted for during a later level pass. After a plan is completed at one criticality level the criticality level is decremented. The abstract plan becomes more detailed as criticality level decreases. The sequence of abstract plans is created differently for each problem depending on the concepts employed.

In summary, this approach utilizes a topdown refinement that is individually constructed to fit each problem in the domain. In this approach it must be possible to assign a criticality ordering to the domain concepts and what is important for one problem must be important for all problems [Stefik et al, 1982].

Case 7: Interacting subproblems

In the previous case it is assumed that similar kinds of decisions should be made at the same criticality level for each problem in the domain. This section explores a reasoning approach based on the least-commitment principle.

This approach requires the following attributes:

1. The ability to know when there is enough information to make a decision.
2. The ability to suspend problem-solving activity on a subproblem when information is not available.
3. The ability to move between subproblems, starting work as information becomes available.
4. The ability to combine information from different subproblems.

The figure above is an example of the least commitment approach used in NOAH [Sacerdoti, 1974]. NOAH

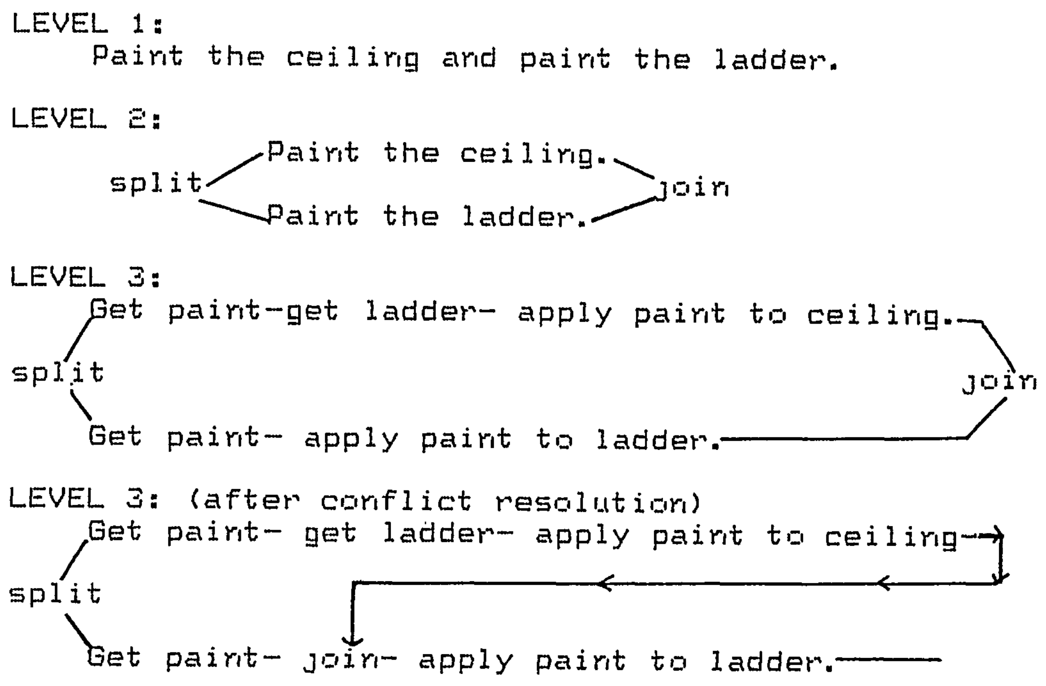

LEVEL 1:
Paint the ceiling and paint the ladder.
LEVEL 2:
split
Paint the ceiling.
Paint the ladder.
join
LEVEL 3:
Get paint-get ladder- apply paint to ceiling.
split
Get paint- apply paint to ladder.
join
LEVEL 3: (after conflict resolution)
Get paint- get ladder- apply paint to ceiling
split
Get paint- join- apply paint to ladder.

Figure 2-5: Example of Planning in NOAH

is a robot planning system that assigns a time ordering to operators in a plan as they are required.

In figure (2.5) NOAH starts with two subgoals which are expanded until a conflict is found. The conflict appears in LEVEL 3. If the ladder is wet it cannot be used to paint the ceiling. At the time the conflict is resolved by altering the plan.

In conclusion, this approach coordinates decision making with the availability of information and moves the focus of the problem solving activity among subproblems. When there are many options and no compelling reasons for choices one cannot utilize this approach.

Case 8: Guessing is needed

Below are listed situations in reasoning when guessing is important:

1. Many problem-solvers need to cope with incomplete knowledge and may be unable to determine the best choice at some stage in the problem solving.

2. A search space may be quite dense in solutions. If solutions are plentiful and equally desirable then guessing is efficient.

3. Sometimes, as in top-down refinement, there is an effectove way to converge the solutions by systematically improving approximation.

The difficulty in guessing is in identifying wrong guesses and recovering from them. Stallman and Sussman [Sussman and Steele, 1980] use guessing in EL, which is a program analyzing electrical circuits.

When analyzing diodes and transistors, EL uses a method of assigned states that requires guessing. For diodes EL has two possible states (On or Off) and three states for transistors (active, cutoff, and saturated). Once a state is assumed EL can use a non guessing method, propagation analysis. After making an assumption EL must check whether the assumed states are consistent with the voltages and currents predicted for the devices.

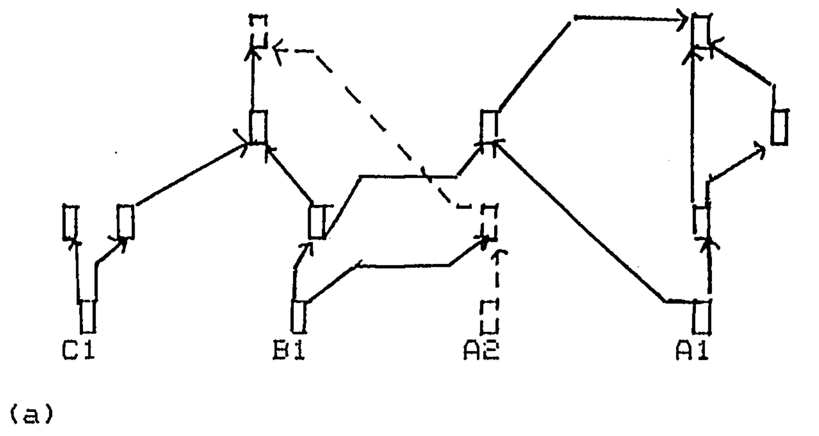
Contradiction is used to detect incorrect assumptions. When a contradiction occurs the assumptions are revised through belief revisions.

In EL an assertion is believed (or in) if it has well-founded support from atomic assumptions, and out when lacking support. If an assertion was out and becomes in it is unouted. In the figure below A1 and A2 are mutually exclusive device-state assumptions. The top portion of the figure demonstrates facts that are in when A1 is in. Arrows indicate support and dotted lines (although part of the data base) indicate what is out. In the bottom of the figure (2.6) A2 is unouted and A1 is outed.

Generally dependency directed backtracking is used in belief revision to recover from incorrect assumptions. The main points are:

1. In the event of a contradiction EL needs to decide what to withdraw. El must decide which assumptions are most unlikely to change.
2. El must redo some of the propagation analysis.
3. Contradictions are remembered so that choice combinations that are found to inconsistent are not tried again.

El employs efficient guessing [Stefik et al, 1982] via dependency directed backtracking to recover from incorrect assumptions.



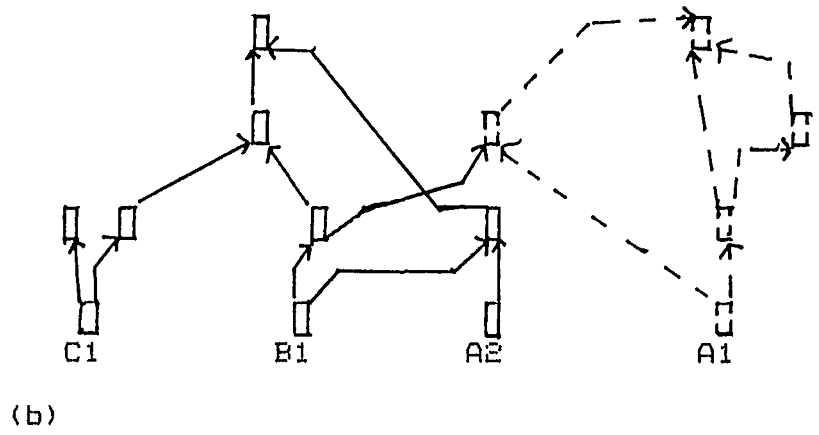


Figure 2-6: Example of Belief Revision in EL

<u>Case 9: Single line of reasoning too weak</u>

In certain instances systems gain power in the use of multiple lines of reasoning in problem solving. The two main purposes for multiple limes of reasoning are to broaden the coverage of an imcomplete search and to combine strengths of separate models.

The expert system HEARSAY [Erman et al, 1980], a system for understanding speech, copes with conflicting demands of searching a large space with limited computational resources by carrying a limited number of solutions in parallel.

A good example of combining strengths of multiple models is the expert system SYN [Sussman and Steele, 1980]. SYN is a program for determining values for components (e.g. the resistance of resistors) in electrical curcuits.

SYN utilized the idea of slices or multiple views of a circuit which corresponds to the idea of equivalent circuits in electrical engineering practice. For example, a voltage divider can be seen as being composed of two alternative slices. One slice of the circuit describes the voltage divider as two resistors and another slice describes it as a single resistor. The program then proceeds in two redundant paths for information to travel in propagation analysis. The strengths of the different models are combined with forward reasoning. When using slices the problem solver must know how to create and combine multiple views.

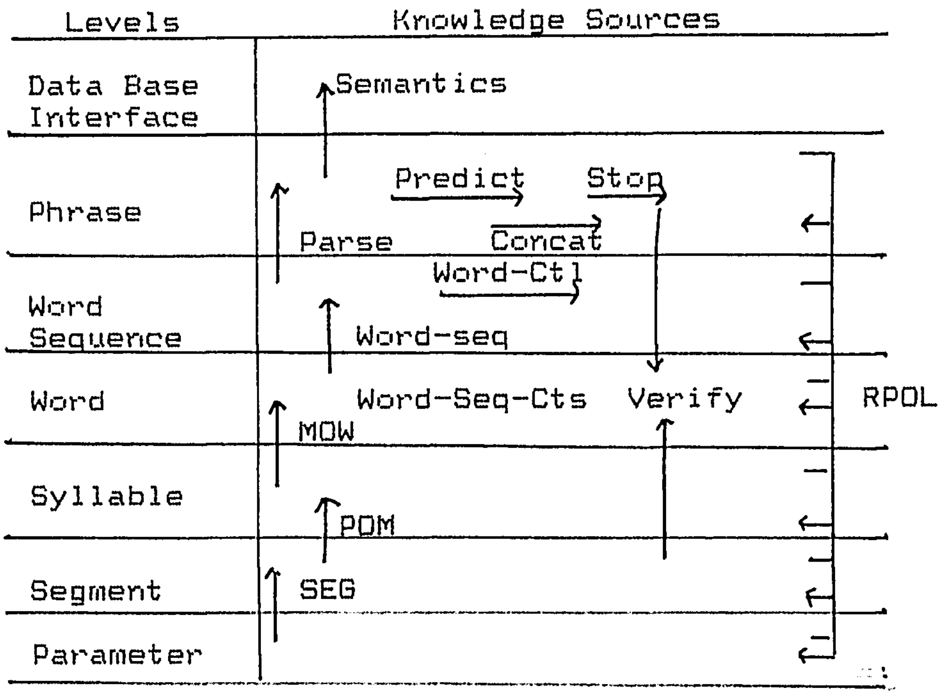
Case 10: Single source of knowledge is too weak

This section explores the use of multiple sources of knowledge in an expert system. HEARSAY reported by Erman [Erman et al, 1980] is once again used as an example.

In HEARSAY-II the knowledge for understanding speech is broken into knowledge sources of information referred to as heterogeneous abstraction spaces. The levels are heterogeneous to match the diversity of the interpetation knowledge (see figure 2.7). The knowledge sources cooperate via an opportunistic scheduler that coordinates the diverse sources of knowledge and adapts to changing conditions of uncertainty in solutions by changing the breath of the search for different hypothesis. The basic mechanism for this is the interaction between knowledge source assigned credibility ratings on hypothesis and scheduler assigned priorities of pending knowledge source activations. Therefore, abiguity between competing hypothesis causes HEARSAY-II to search with more breath, and to delay the choice among competing hypothesis until more information is available.

Case 11: General representation methods are too inefficient

As knowledge bases get larger, the efficiency penalty incurred by using uniform representations can become significant. One change in the representation of knowledge that is explored for expert systems is



The knowledge sources are as follows:
Semantics: generates interpretation for the information system.
SEG: digitizes the signal, measures parameters, produces labeled segmentation.
POM: creates syllable-class hypothesis from segments.
MOW: creates word hypotheses from syllable classes.
Word-Ctl: controls the number of hypotheses that MOW makes.
Word-Seq: creates word-sequence hypotheses for potential phrases.
Word-Seq-Ctl: controls the number of hypotheses that Word-Seq makes.
Predict: predicts words that follow phrases.
Verify: rates consistency between segment hypotheses and contiguous word-phrase pairs.
Concat: creates a phrase hypothesis from a varified contiguous word-phrase pair.
RPOL: rates the credibility of hypotheses.

Figure 2-7: Levels and Knowledge Sources in HEARSAY-II

knowledge compilation. 37 This technique transforms one representation of knowledge into another representation that can be used more efficiently.

Burton reported a system [Burton, 1976] for taking ATN grammars and compiling them into executable code. [Stefik et al, 1982] The compiled grammar could be executed to parse sentences much more rapidly than previous interpreter-based approaches.

The promise of knowledge compilation is to make it possible to use general means for representing kowledge when an expert system is being built and debugged. Then the compiler can be applied to make the knowledge base more efficient. In addition, as hardware is changed or as trade offs in representation become better understood, the compiler can be modified to represent knowledge efficiently.

In summary, the first case considers an expert system of a very simple architecture that requires small search space and data and knowledge that is reliable and constant. In successive cases the following attributes of an expert system are considered: unreliable data, time varying data, and big solution space. The cases of an expert system with a big solution space, but requiring organization to accomodate other complex structures are developed. The organization of an expert system relects the availability of search space and the characteristics of the knowledge base and data.

2.5 Software Tools

The development of expert system software tools is an area requiring increasing attention. Currently much work in knowledge engineering focuses in developing computer programs that aid scientists with complex reasoning tasks. However, building these programs is a time consuming task [Tanaka, 1982].

One of the techniques used in an attempt to harness the power of expert systems more efficiently is by building up the knowledge base of such systems with special purpose programming languages.

The best known of these is Lisp, a language developed orignally as a means of proving correctness in programs and taken up by the artificial intelligence community as its major tool [McCartney, 1982]. Lisp provides a rich, interactive editing and debugging environment. More fundamentally, Lisp removes the distinction between programs and data by treating the rules and heuristics in the knowledge base sometimes as data to be reasoned about and sometimes as code to be executed. [Davis and Lenat, 1982].

A lesser known example is Prolog, a software language designed for artificial intelligence applications, such as, expert systems [Goodall, 1983]. Prolog stands for PROgramming LOGic. It is based on predicate calculus [Rundle, 1982] and is very different from the standard type of programming language or notation.

Prolog was originally developed at the University of Marseille at the beginning of the 1970s as a means of using computers to understand so called 'natural' languages. The study of natural language is closely allied with the study of artificial intelligence. [McCartney, 1982] Expert systems are a prime example of the type of the type of application to which Prolog is well suited.

The rules and conditions which comprise the knowledge base of an expert system can be easily represented by Prolog's data structuring facilities. [Goodall, 1983] A Prolog program consists of a number of rules or facts about a subject. Once defined you can ask prolog questions about the subject and it will attempt to answer them [Rundle, 1982]. The inference engine of an expert system needed to manipulate these rules and conditions can make use of the language's own inference mechanism (which does not have to be the same as that of the expert system).

Writing in Prolog is quite different from writing in a traditional algorithmic language. Instead of asking 'what is the algorithm that will solve my problem?' the programmer asks 'what are the facts and rules which describe my problem?' Having determined the facts and rules, the programmer can than state them very naturally in Prolog (this is where the basis in logic comes into play) and the problem is given a formal specification. At this point the programmer changes the way he views the program and considers it as a set of procedures which when executed, will perform a controlled deduction through the logic statements. To some degree the details of how this deduction takes place can be left unspecified by the programmer, since this is handled by the language's inbuilt inference mechanism [Goodall, 1983].

```
trained-on(adams,mx01).   adams is trained on an mx01
trained-on(brown,mx01).   brown is trained on an mx01
trained-on(brown,mx02).   brown is trained on an mx02
trained-on(carter,mx03).  carter is trained on an mx03
owns(avis, mx01).         avis owns an mx01
owns(avis, mx03).         avis owns an mx03
owns(bbc, mx01).          bbc owns an mx01
owns(cook, mx02).         cook owns an mx02
owns(cook, mx03).         cook owns an mx03
```

Figure 2-8: PROLOG data base

Consider, for example, (see figure 2.8) a number of servicemen (adams, brown, and carter); a number of machines which they service (mx01, mx03, mx02); and a number of customers who own one or more machines (avis, bbc, cook). The above Prolog database describes the situation in detail. All the above are 'facts' which in this case do not contain variables. The general form of a Prolog definition of a relationship or assertion is: <relationship> [<subject> <object>].

```
    can-call-on [S,C]:- owns [C,M], trained-on [S,M]
```

The above is a prolog 'rule'. The symbol :- means 'if' or 'provided that'. S,C and M are variables becase they start with an upper case letter. The rule says 'A serviceman S can call on a customer C provided that C owns some machine M and that S has been trained on M.' The rule can be used to locate all Cs, given an S, or all Ss given a C, or all valid combinations of S and C.

The above facts and the single rule constitutes a very simple Prolog program that purely describes the serviceman/machine/customer world. If this were in a Prolog database one could ask questions of the database as in figure 2.9.

Questions are preceded by a '?'. [Goodall, 1983]

Although, little commercial use is being made of Prolog currently, it is likely to be only a matter of time before it will become used in the world of business in software departments as a tool to build expert

```
?-trained-on(S,mx01).       Who is trained
                            on an mx01?
S=adams                     adams is
More(y/n)?y
S=brown                     and brown is
More(y/n)?y
no
?-owns (bbc,M).             What machines
                            does bbc own?
M=mx01                      an mx01
More(y/n)?y
no                          and no more
?-can-call-on(S,avis).      Which servicemen
                            can call on avis?
S-adams                     adams can
?-can-call-on (brown,C).    On which customers
                            can brown call?
C=avis                      on avis
More(y/n)?y
C=bbc                       on bbc
```

Figure 2-9: Sample Questions of Database

systems. [McCartney, 1982]

Some other software tools besides special purpose programming languages are tools that have been developed to aid the design process of the knowledge base of an expert system.

EXPERT and EMYCIN (essential MYCIN) assist in the development of production systems [Mizoguchi, 1982]. EMYCIN is not itself an expert system- it is a means of building such systems and one way of getting around the problem of setting up the rule database.

Essential MYCIN is the central core of MYCIN, and is used as a domain independent system to develop other rule-based systems. EMYCIN has been applied to expert systems in other medical areas, such as pulmonary disorders (PUFF) and psychiatry (HEADMED) as well as structural analysis (SAUM).

According to researchers at Stanford University, the most recent application of EMYCIN took only 20 hours to build. In a recent report published by "Pergamon Infotech on Machine Intelligence", the system CLOT which was developed for diagnosing disorders of the blood coagulation system- "was constructed as a joint effort by an experienced EMYCIN programmer and a collaborating medical student. Following approximately 10 hours of discussion about the contents of the knowledge base, they entered and debugged in another 10 hours the preliminary knowledge base of some 60 rules using EMYCIN" [McCartney, 1982].

A software product called AL/X (standing for 'advice language') has been developed by Michie. It is designed for programming expert systems and was developed in conjunction with BP (a British company) at Dyce. Its successor, coded in the language C, will be available in the near future and run on any Unix or Unix-like operating system. Available at that time will be a component, Intelligent Terminals' rules from examples system, that is designed to automate the compilation of rules from examples. It is intended to increase the productivity of human experts.

Michie contends that the bottleneck in building expert sytems is the acquisition from the human expert of a huge number of rules that are only partially and imprecisely accessable to the conscious mind. This system will operate as if one were teaching an apprentice. Examples will be fed in and a rule will be produced automatically to encompass them.

The immediate application would be an expert system designed for testing electronic equipment with fault tables. Long term application would be computer vision. In the not too distant future this system will be available as a floppy disk for the Apple computer [Burkitt, 1982].

In developing expert systems, KRL, FRL, UNIT and RLL, are available as general purpose representation systems for a knowledge base. [Mizoguchi, 1982]. More recently, AGE (Mizoguchi-17) has been proposed as a tool for designing general purpose knowledge base systems.

If the tools for knowledge base systems are suitable to the problem domain, the necessary task of the designer is the task of selecting the best tool among them and formulating the problem in terms of the specifications of the tool. The design process is highly dependent upon the software tool that is applied to the problem [Mizoguchi, 1982].

Another approach to software tools, is the concept of SAUI, the self-adaptive user interface. This tool would be under the category of what Waterman [Waterman, 1978] describes as a small program that sits between the user and the system. The user interacts with the SAUI and the SAUI is capable of performing a variety of tasks for the user by interfacing the user with the expert system. The SAUI provides the user with help in learning and using complex expert systems. [Innocent, 1982] Software tools, such as, SAUI are in the planning stage.

A system called Multi-Layered Software Environment (MLSE) has been proposed for providing a designer of an expert system with a wide variety of design alternatives in software tools derived from artificial intelligence technology. It is a collection of module packages for building the components of knowledge base systems. This system emphasizes a layered approach to building the software environment as a basis for developing a knowledge base system [Mizoguchi, 1982].

A British company, SPL, has introduced a new software package called SAGE which it claims is the first general purpose development program for expert systems. In other words it is software that will enable the user to produce their own expert system on any subject they like.

The purpose of the software tools previously mentioned are to alleviate the burden of writing an expert system and allow the user to implement a system with less instructuion that was available in the past.

2.6 Applications

Expert systems are regarded as a major computing development. They are the first practical implementation of research into artificial intelligence. One of the reasons for their success is that they are not apparently 'clever' and do not make human beings feel inferior [McCartney, 1982]. Expert systems are convivial to the extent that they make previously scarce expertize available to the user. They are congenial to the extent that they interact with the user in his or her language and offer assistance in a mode that allows the user to retain decision making perogatives. The expert system, moreover, contains knowledge in a formalism natural and understandable to the user. The system contains an explanation capability to explain the 'why and how' of its rasoning [Bendifallah, 1982]. An expert system has the ability to accept rules and experience concerning a specific domain and make deductions about that domain

[McCartney, 1982].

Virtually any problem domain is suitable for solution by an expert system provided the knowledge necessary for solving the problem domain can be put into rule form. However, it is necessary to remember that if a problem domain generates less than about 10 rules, it is probably not worth using an expert system, since a human can solve it just as efficiently [Winfield, 1982].

Recent domains providing practical applications for expert systems are medical diagnosis and therapy; equipment failure diagnosis; computer configuration; chemical data interpretation and structure; experiment planning; speech and image understanding; oil field services; military needs; mineral exploration; military threat assessment and targeting; crisis management; science; advising about computer system use; training teaching; and air traffic control [Feigenbaum, 1982].

Some existing practical applications of expert systems are listed below.

Application Area	Name	Comments
MIneral exploration	Prospector	Interprets surface geology.
Translation of meteorological bullitens	TAUM	Translates meteror-logical bulletins from English to French.
Materials handling		Microcomputer ES to help select

		handling techniques.
Mass spectral	DENDRAL	First ES. interprets mass spectra (Chem. analysis).
Medical test analysis	PUFF	Diagnoses pulmonary diseases.
Plant pathology	AQ11	Exceed human diagnosis of soyabean diseases.
Oil platform faults	AL/X	Diagnosis automatic shutdowns.
Medicine Psyco		Diagnosis Dyspepsia
Tax advice	TAX ADVISOR	Advice on capital transfer tax.
Science	CONCHE	Aids scientific theory formation.
System design	R1	Configures DEC VAX/780 Computer systems.
Fault Diagnosis	CRIB	Diagnosis computer hardware and software faults.
Medicine	MYCIN	Diagnosis and drug treatment.
Education	GUIDON	Tutor improves students diagnostic skills.

Particularly noteworthy are MYCIN, AQ11, PROSPECTOR, R1 and DENDRAL. Medical consultation systems are a major application of artificial intelligence research [Kaihara and Koyama, 1982]. MYCIN, an expert system that diagnoses blood diseases and selects antibiotic therapy for bacteremia has been mentioned previously in this paper. [Chester, 1982] Developed at Stanford University, it is one of the earliest and simplest expert sytems. The MYCIN system contains about 450 rules which are used for diagnosis. It has been developed further by the addition of 200 or more rules to be used as a teaching aid and covering both facts and problem-solving strategies. The extra rules cover methods of guiding dialogues with the students and presenting strategies to the students. [Rundle, 1982] AQ11 is a system which has a 99% success rate in diagnosing soyabean diseases and is now used by the top human experts. [Ellis, 1983]

Dendral is another well known system that originated at Stanford University. It is designed to determine the molecular structure of organic compounds from their chemical formulas using mass spectrograph and nuclear resonance data. [Chester, 1982] The program has been used to establish new molecular structures [Rundle, 1982].

One of the best known engineering expert systems is PROSPECTOR [Boothroyd, 1982]. PROSPECTOR is capable of mapping underground ore deposits from observed surface features [Ellis, 1983]. A company SRI International was commissioned by the United States Geological Survey and US National Science Foundation to develop PROSPECTOR [Rundle, 1982]. This system gives geological advice to mineral companies looking for the likeliest sites to find copper and molybdenum. [Boothroyd, 1982]

PROSPECTOR contains rule-based models of different ore deposits which can evaluate the likelihood of finding a particular type of ore in a geological district, and select the best drilling position on the exploration site [Rundle, 1982]. The 1600 or so rules comprising PROSPECTOR's knowledge base were developed by interviewing a number of geologists who were recognized experts in their field, and building up the associations between observable evidence and the likely underlying geological structure. Moreover, PROSPECTOR is capable of giving details of the rationale for conclusions reached and suggesting which data are most valuable for further exploration [Rundle, 1982].

PROSPECTOR has correctly contradicted human experts. The US company Fairchild was considering exploration for a deposit of the rare element, Molybdenum, on a site that its advisors told them was not worth the investment. PROSPECTOR said the opposite ard was proven to be correct [Boothroyd, 1982].

Digital Equipment [McDermott, 1982] has pioneered the use of expert systems for working out the demands of its customers and turning them into a machine configuration. R1 designs complex computer systems. The system has been used extensively for this purpose on their latest range of VAX computers. [McCartney, 1982] The system has about 800 rules governing the configurations, together with a database describing about

400 components. [Rundle, 1982] Eventually, the configuration produced by the computer will be fed automatically to the factory where manufacturing will start under computer control [Boothroyd, 1982].

Presently, expert systems can store and amplify rare specialist expertize and make it more widely accessable. Expert system techniques offer a route to solutions to high software development costs, incomprehensibility of programs and the inability of the ordinary user to intervene without a programmer to help.

A problem in today's world is the shortage of human expertize. It is scarce and expensive. Human experts are fallible and their compacities limited and, of course, they are mortal. In contrast, expert systems are capable of reducing skill shortages. They are widely distributable, easy to run, duplicate and upgrade. Expert systems have the capability of excelling humans in complex problems, and they cannot resign or die.

Some of the management concerns of the 1980's are the acquisition of competent management, too little time to solve problems, an overload of information, lack of trained personnel, and the availability of material resources. Expert systems are now capable of policy analysis and strategy, of augmenting management skills, and formulating and solving existing problems. Expert systems are decision oriented. They can educate personell, assist in exploration of resources and cut risks and costs of management.

Previously people complained about the difficulty of conversing with computers, the difficulty in understanding computers and the expense in modifying and developing computer software. Expert systems bridge the man-machine gap. They talk in user language, can explain reasoning, and are trivial to modify. They are the best route in encouraging progress in automatic programming.

There is concern that present complex computer systems are dangerous because, for example, they can emit false missile alerts, allow mistakes, such as, occurred at 3-mile-island and are hard to monitor in air traffic control. Expert systems return human control by providing a 'human window' that allows the user to comprehend the system and enables faults to be spotted and disasters averted. [Ellis, 1983]

In the future, if successfully developed, the fifth generation computer sytems will be excellent vehicles for expert systems applications [Feigenbaum, 1982]. Recognizing the importance of knowledge based industries in the 21st century, the Japanese are two years into a ten year program to develop 5th Generation Computers. These computers will go radically beyond all previous computers and put useable information technology at everyone's elbow. Details released in Tokyo in October, 1981, to computer experts from Western countries detailed three key elements of design.

1. Very large scale integrated components- high power at low cost on small chips.
2. Distributed processing- distributing computer via telecommunications and
3. Expert systems making computers behave more like people, and also leapfrogging current software quagmire. [Ellis, 1983]

The Japanese, are committed in their Fifth Generation project to having systems with over 10,000 rules within the decade. [McCartney, 1982] The social and economic goals of this project are ambitious and would include Japan providing world-wide leadership in information technology [Parrott, 1983].

Fifth generation computer expert systems will be primarily symbolic manipulation systems. They will be knowledge processors with arithmetic capabilities. They intend to meet the major commercial demands of personal and professional expert systems from the period of 1990 to 2000. Much of today's software will appear on the chip as hardware in these systems. The software ideas of today are the seeds of the big ideas for the Fifth Generation Computer expert systems [Feigenbaum, 1982].

Expert systems have powerful implications for managers, professionals and organizations. Expert systems are a human face of information technology and will find an application in every sector and level of modern economy.

Expert systems will change the ways managers and professionals operate by their ability to call an expert system for decision making. Expert systems will have increased capability in the future and reduced response time. In the professions, top experts will find a new channel to market their skills, allowing them more time for research and checking assisted solutions. Lesser experts, hopefully, will see expert systems as a better type of 'manual'.

Collectively, these efforts could radically alter the performance of organizations. If expert systems and advanced information technology are to be introduced beneficially, a coordinated strategic response may be required. The advantages of expert systems applications can be enormous if the applications, timing, investment profile and employee relations are all considered.

3. Major Research Issues of Expert Systems

This chapter discusses some major research issues of expert systems. They are heuristics, knowledge representation and knowledge acquisition.

3.1 Heuristics

Builders of expert systems attribute the impressive performance of their programs to the body of knowledge they contain: a large network of facts and a large array of heuristics. Heuristics are informal, judgmental 'rules of thumb'.

Heuretics who study heuristics extract heuristics from experts. They decide when the existing corpus of heuristics needs to be augmented. They represent heuristics within the knowledge base, and evaluate the worth of a particular heuristic in a program in troubleshooting an expert system built with heuristic rules.

Researchers of heuristics study the origin of heuristics and the source of the power of heuristics. The source of power of heuristics can be seen as a two dimensional continuity [Lenat, 1982]. If a heuristic H was useful in situation S, then it is likely that heuristics similar to H will be useful in situations similar to S. If one were to compute the function

APPROPRIATENESS (Action, Situation), that function would be continuous in both variables, and would vary very slowly. Although, according to Lenat, appropriateness can be measured in many ways (such as, efficiency and comprehensibility) and situations can vary (with difficulty, time, importance and subject matter), it is often useful to behave as though the function appropriateness (action, situation) exists and is continuous. If one does so then one is following a heuristic.

One must consider the continuity, stability and observability of a domain in determining whether an expert system utilizing a heuristic search will be of assistance. If data is not observable and cannot be gathered then heuristics cannot be formed and evaluated. If the environment is not continuous and canges abruptly, the heuristics may never be valid. If the changes are continuous but too rapid to be stable then the heuristics may have too short a lifetime before becoming useless.

According to Lenat's [Lenat, 1982] empirical results from AM, an expert system designed to discover mathematical concepts and conjectures, new heuristics arise from three sources: specialization, generalization and analogy. Specialization of existing, more general heuristics can provide one or more new heuristics. This can occur, for example, when matching more specified observed data to a template in a computer program or it can occur when noting an exception to a genral heuristic, and therefore formulating a higher precedence heuristic.

Generalization of existing, more specialized heuristics can occur. Commonly an abstraction of a heuristic applied in a more specific area of a program can be applied more generally to a greater domain in the expert system. Analogy to existing heuristics and to successful acts of creating new heuristics is the third origin of heuristics. In AM, for example, Lenat was able to look for examples of concept C before trying to prove any theorems about C.

Some examples of domain heuristics in AM are illustrated in figure 3.1 below.

```
H1:  IF:    A X A -> B,
     THEN:  define G: A-> B as G(x) = F(x,x)

H2:  IF:    F:A->B, and there is some extremal
            subset b of B,
     THEN:  define and study F^{-1} (b)
```

Figure 3-1: Two Heuristic Rules

Heuristic H1, says if a function F takes a pair of A's as arguments, then it's often worth the time and energy to define G(x)=F(x,x), that is, to see what happens when 'F's arguments coincide. If F is multiplication, for example, this new function 'G' becomes squaring. Heuristic H2 says to investigate the inverse image of known b. If 'F' is intersection, H2 says it's worth considering pairs of sets which map into extremal kinds of sets (e.g. extremely small sets, such as the empty set). This heuristic could lead to defining the relationship of two sets having empty intersection or disjointness.

An expert system, EURISKO [Lenat, 1982], which is an extension of the previously mentioned AM, is a program built with heuristic rules and is capable of discovering new heuristics as well as new mathematical concepts. Below is an example of three heuristics in EURISKO capable of working on heuristics as well as math concepts domain. Meta-heuristics are heuristics which inspect, gather data, modify and synthesize other heuristics. Their counterpart are domain heuristics that define what we mean by a particular domain of knowledge (i.e. mathematic concepts), and are object level heuristics.

The first one says that if some concept f has always led to bad results, then f should be marked as less valuable. Concepts in EURISKO are knowledge represented by the frame method. If a mathematical operation, like Compose (which refers to mathematically composing two

```
H12:  IF:    the results of performing F have
             always been numerous and worthless,
      THEN:  lower the expected worth of F.

H13:  IF:    the results of performing F are
             only occasionally useful,
      THEN:  consider creating new specializations
             of F by specializing some slots of F.

H14:  IF:    a newly-synthesized concept has slots
             that coincide in value with those of
             an already existing concept
      THEN:  the new concept should be destroyed
             because it is redundant.
```

Figure 3-2: Heuristic Rules in Eurisko

functions), did not ever lead to any good new math concepts, then this hueristic would lower the number stored on the 'worth' slot of the compose concept. Likewise, if a heuristic, for drawing diagrams never was utilized then its 'worth' slot would be decremented.

The second heuristic H13 says that if some concept has been cocasionally useful and frequently worthless, then it is worthwhile to investigate specialized versions of that concept. H13 was utilized in AM, for example, to find new specializations of the compose concept to create a function, composition of a function with itself. In EURISKO H13 was further developed to apply H13 to heuristics. In fact H13 once applied to itself. One of the specializations resulting was heuristics which demand that it has proven itself at least 3 times.

Heuristic H14 enables EURISKO to eliminate redundant concepts created perhaps by other heuristic rules. The heuristics of EURISKO are capable of operating on each other (and themselves) to synthesize new heuristics.

The field of heuretics is a promising one for Artificial Intelligence to investigate in helping one to understand and construct expert systems. The power of heuristics lies in behaving as though appropriateness (action, situation) were time invariant and continuous in both variables. Heuristic search is appropriate when modeling domains that are observable, stable and continuous. Heuristics originate from generalizing other heuristics, specializing other heuristics, and finding analogies to other heuristics.

EURISKO demonstrates that there is not a need to distinguish between object level heuristics and meta-heuristics. Continued research in heuristics will hopefully provide new ways to improve and understand expert systems.

3.2 Knowledge Representation

Expert systems are unique in that they use an ordered set of task-specific rules to solve problems in a way similar to how an expert in a particular technical field might do it. [Webster, 1982] These set of rules lay down the relationships and correlations between information segments in the knowledge base. Determining the best of many possible ways of representing this knowledge and the rules connecting different items in the knowledge base becomes an important consideration [Gowans, 1982]. Much research work is currently being pursued into ways of representing knowledge in expert systems [Winfield, 1982].

1. Logic
2. Procedural representations
3. Semantic networks
4. Production systems
5. Direct (analogical) representations
6. Semantic primitives
7. Frames

Several methods of representing knowledge are currently used. Logical deduction by using predicate calculus is one method [Barr and Feigenbaum, 1980]. In a Procedural representation knowledge is accessed by direct explicit calls of each procedure [Winston, 1977]. When using the semantic networks method semantic attributes are included in the representation of a rule. The attributes connect the rule to other rules, thereby, more explicitly defining the attributes [Catanzarite, Greenburg and Bremermann, 1981]. In a production system the knowledge is represented by a series of productions [Winston, 1977]. And sometimes, as in direct (analogical) representations whole systems can be powerful metaphors which facilitate a problem solution through stong analogical features [Winston, 1977]. When semantic primitives are used each condition can be considered a semantic token, upon which other information can be attached [Leith, 1983]. Finally, in frames the knowledge is factual and can be represented by methods extending from simple tables to sophisticated frame systems [Winston, 1977].

The method which has been used in the majority of the more common present day expert systems is the production system. A production system consists of a number of rules where each rule is of the IF...THEN... type. Sometimes these rules are referred to as situation action rules; that is IF some situation occurs THEN some action is performed [Winfield, 1982].

R1 [McDermott, 1982] is an expert system using a production system to represent kowledge. R1 currently has 772 rules that enable it to configure the VAX-11/780 computer system. An English translation of a sample rule is shown in figure 3.3.

The first condition of this rule indicates that the subtask in which the rule is relevant is the distributing of massbus devices among massbuses. The remaining five conditions specify one of the sets of constraints that

```
DISTRIBUTE-MB-DEVICES-3
 IF: The most current active context is distributing
         masbus devices
     and there is a single port disk drive that has not
         been assigned to a massbus
     and there are no unassigned dual port disk drives
     and the number of devices that each massbus should
         support is known
     and there is a massbus that has been assigned
         at least one disk drive
     and that should support additional disk drives
     and the type of cable needed to connect the
         disk drive to the previous device on the
         massbus is known
 THEN: assign the disk drive to the massbus.
```

Figure 3-3: R1 Sample Rule.

must be satisfied within this subtask in order for a disk drive to be assigned to a massbus. One of the single port disk drives on the order is assigned to one of the massbuses when an instantiation of the rule is executed.

Various properties of production systems, which have contributed to the popularity of this form of knowledge representation have been listed by Davis and King [Davis and King, 1977]. They include their modularity, the driveness and openess of control, the constrained format of the rules, and that new rules can be incorporated easily [Winfield, 1982].

Production systems seem to be appropriate for domains whose methodologies are modular and subject to frequent alteration. In contrast procedural systems seem more appropriate for domains with well-defined and integral methodologies.

However, when a measure of feasibility is required in an expert system, the procedures representing these methods can be datadriven procedures which are allowed to edit their own driving data in a learning system whose methodology is capable of changing in detail as experience is acquired [Smith and Bowen, 1982].

IIDA (Individualized Instruction for Data Access System) is an example containing a procedural knowledge representation. It is a system [Meadow, Hewett, and Aversa, 1982] that serves as intermediary for users in performing a complex task on another computer. Another example is MAPLE (Microprocessor Application Expert).

MAPLE is a prototype expert system bying developed by Bowen. It is an interactive system which assumes the rule of a consultant expert in the field of hardware design for microprocessor applications. Because design of microprocessor applications using board level components is a field for which standardized methodology is defined, MAPLE is being implemented as a procedural system. MAPLE's knowledge of its domain is composed of three parts: its methodology of application design, its access to information about the components needed in microprocessor systems and its experience of past applications. The methodology is encoded as procedures. The approach of MAPLE to the design of microprocessor applications is represented as a set of data-driven procedures. Therefore, its methodology can change in detail as it acquires experience. Component information and application experience are stored in data files and are therefore can be extended [Smith and Bowen, 1982].

Another method of representing knowledge is described as a semantic network. A network provides a particularly rich structure for entering detailed relationships and descriptors in the domain model. Wallis and Shortliffe have designed a prototype system to expand explanatory power for medical expert systems. They describe their system as having a semantic network knowledge representation.

Figure 3.4 demonstrates a sample section of network from this program showing object, parameter, value and rule nodes.

Dotted lines indicate the following rule

```
IF PARAMETER-1 of OBJECT-1
   is VALUE-1, and
   PARAMETER-2 of OBJECT-1 is VALUE-4
THEN conclude that PARAMETER-4 of OBJECT-3 is VALUE-7
```

Object nodes are arranged hierarchically, with links to the possible attributes (parameters) associated with

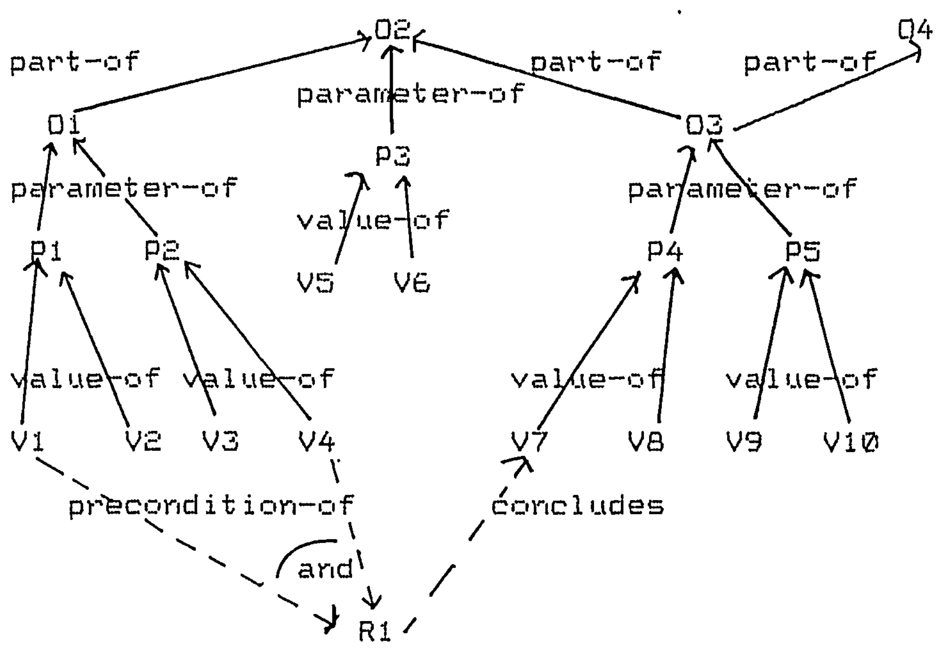


Figure 3-4: Sample Section of Network.

that object. The parameter nodes are linked to the possible value nodes, and rules are themselves represented as nodes with links that connect value nodes. These relationships are summarized below in figure 3.5.

The certainty factor refers to the model developed for the MYCIN system. A certainty factor can have a value ranging from -1 to +1. Ask first/last (figure x) is a property that controls whether the value of a parameter is to be requested from the user before an attempt is made to compute it using inference rules from the knowledge base. The text justification of a rule is available for when the system builder wishes to provide a

Type of Node	Static Information (Associated with Node)	Dynamic Infor. (Consultation Specific)
Object node	part of link (hierarchic) parameter list	
Parameter node	object link value-node list default value text definition	
Value node	parameter-node link precondition-rule list conclusion-rule list importance complexity ask first/last	contexts for which this value is true certainty factor explanation data ask state
Rule node	precondition list (Boolean) conclusion certainty factor rule type complexity text justification	explanation data

Figure 3-5: Relationships of Nodes.

brief summary of the knowledge underlying that rule.

In order for the system to provide adequate explanations, the semantic network associates a measure of complexity with the inference rules and the concepts about which they are concluding. A measure of importance is associated with concepts because some concepts are key ideas in a reasoning chain and should be maintained regardless of their complexity [Wallis and Shortliffe, 1982].

In this particular program the semantic network provides a rich structure for enhancing the explanation capabilities of reasoning programs for medical consultation.

An expert system that represents knowledge in semantic primitives has been reported by Leith. ELI (Expert Legislative Information) system opeates in the field of legislation. The knowledge base of ELI contains individual conditions that can be physically shared by rules. The advantage of this approach is that when one condition is common to many rules - a substantial amount of storage can be saved. More importantly, each condition can be treated as a semantic token representing one chunk of causal knowledge. Similar in purpose to systems with semantic network reprsentations the ELI system can tell the user where a condition was extracted from by the expert. Each conditon in this semantic representation is a semantic token upon which other information can be attached. The user is provided with more information then would otherwise be possible.

Included in information that can be attached to a semantic token are notations which can be associated both with production rules and individual conditions themselves. Each condition can be, for example, associated with a specific piece of precedent (e.g. from the Law Reports), or a section of legislation.

As parts of an individual rule the conditions represent one cause for that rule having (or not having) been triggered. Moreover, as common elements of more than one rule, they represent common aspects of the pattern of triggering those rules. Causal links in ELI are from conditon to following condition to eventual goal.

Figure 3.6 illustrates a two dimensional representation of this aspect. Nodes represent conditions and arcs represent links.

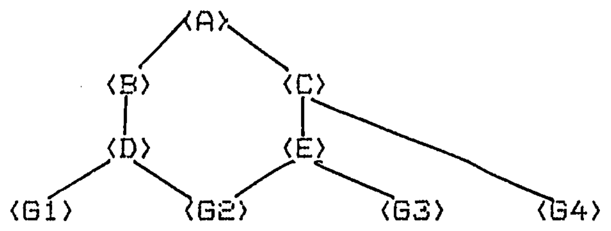


Figure 3-6: Knowledge Representation of ELI.

Thus by using a semantic primitive knowledge representation that is hierarchically structured, an attempt as been made to present a rich source of semantic information to the user. The semantic representaion provides help in judging the truth of each condition, and also provides extra information that can be extracted from the system by the user [Leith, 1983].

Two knowledge representations that are most suited for expert systems with reliable data and knowledge are the logic and direct (analogical) representations.

An example of a logic reprsentation would be using predicate calculus to represent knowledge in a list of inferred facts [Barr and Feigenbaum, 1980]. See the example below.

```
[On [block1 block2]
[On [Block1 Block2]]
[Not [On  Block2 Table1]]
```

TAXMAN-I is an example of the use of direct representations. Analogies are in the form of templates that match a set of particular factual situations [McCarty et al, 1979]. The TAXMAN-I system operates in the problem domain of taxation of corporate reorganizations.

This area of the law is well suited to an analogical knowledge representation. The factual situation described in TAXMAN, though complex, can be described fairly completely using a manageable set of primitive terms. And the operative legal concepts, such as, the definitions of a Type B, a Type C and Type D reorganization have a statutory structure that is articulated in unusual detail.

Constructed on a factual foundation, the TAXMAN-I system consists mostly of propositions of the form: 'Phellis owns 250 shares of the common stock of the Delaware corporation', 'the Delaware corporation transferred its assets to the New Jersey corporation',etc. The higher level conceptual structures of this system take the form of logical templates. A 'logical' pattern is 'matched' to the lower level factual network in both abstraction and expansion process.

TAXMAN-I's knowledge does not go beyond a tidy world of formal financial rights and obligations. The domain of corporate reorganization is an unusually artificial domain well suited to this representational technique. The concepts are treated as static structures applied timelessly to facts.

The final method considered for representing knowledge is a frame representation. The frame method of representation is being used more frequently then heretofore. Although an individual frame may be considered by itself to be a template, Minsky [Minsky, 1974] outlines how in a frame representation a set of frames can be connected by pairwise 'difference descriptions' into a 'similarity network'. The similarity network can then be aggregated into a system of conceptual 'clusters' that are loosely centered around their respective conceptual 'capitols'.

These ideas have been implemented in the AM expert system [Lenat, 1982]. Frames are used to describe mathematical concepts. Specific production rules and procedures are attached to each concept frame. Each concept consists of a collection of properties or 'facets' of the concept called slots. Below is an example of a concept in AM:

```
NAME: Prime Numbers
DEFINITIONS:
       ORIGIN: Number-of-divisiors-of(x) = 2
       PREDICATE-CALCULUS: Prime(x) <=>(for all z)
          (z x=>(z = 1 z =x))
       ITERATIVE: (for x>1): for i from 2 to  x, i   x
EXAMPLES: 2,3,5,7,11,13,17
       BOUNDARY: 2,3
       BOUNDARY-FAILURES: 0,1
       FAILURES: 12
GENERALIZATIONS: Nos., nos. with even no. of divisors
SPECIALIZATIONS: Odd Primes, Prime Pairs, Prime
          Uniquely-addables
CONJECS: Unique factorization, Goldbach's conjec.,
          Extrema of No-of-divisors-of
INTU'S:  A metaphor to the effect that Primes
          are the building blocks of all numbers
ANALOGIES:
          Macimally-divisible numbers are converse
          extremes of Number-of-divisors-of
          Factor a non-simple group into simple groups
INTEREST: Conjectures tying Primes to Times,
          to Divisors-of, to related operations
WORTH: 800
```

Figure 3-7: Frame of AM

In summary, there are many ways of representing knowledge in an expert system. Although the production system method is most commonly used in present day expert systems, the frame system is gaining increasing has attention. This section has attempted to provide illustrations implementing various knowledge representation methods.

knowledge_acquisition

3.3 Knowledge Acquisition

To achieve high performance it is necessary to acquire and maintain a large knowledge base in an expert system. Because it is a formidable task to put an initial knowledge base together using a suitable representation, generality becomes important in the methods for constructing and maintaining large domain-specific knowledge bases [Davis and Lenat, 1982]. Moreover, to enable humans to work satisfactorily in the modern world it then becomes necessary for them to keep the knowledge base of the expert system current and accurate [Winfield, 1982].

Initially, the problem must be analyzed and relevant knowledge extracted so that it can be put into a series of rules [Winfield, 1982]. When considering acquiring knowledge one needs to know what kind of knowledge is required and how much. Are there 10 facts or 1000 facts?

Are most cases covered by a dozen basic methods? There is a tendency to grossly overestimate. Once it is determined that a task is reasonably complicated either a domain expert can construct the knowledge base or a knowledge engineer can collaborate with the domain expert in an attempt to isolate the rules before constructing the knowledge base.

The isolation of rules is a slow process and Michie has said that approximately two rules per week can be built into the knowledge base. However, members of SRI in the United States consider that rules can be extracted from experts in the field considerably faster. In either case it appears that the last few rules take the longest time to extract.

In the future the Japanese propose building large knowledge bases of 20,000 rules (existing expert systems generally use hundreds of rules only). It will become necessary to find some way of speeding up and automating the way rules can be built [Winfield, 1982].

Besides the formidable task of putting an initial knowledge base together, in open ended problem areas, such as medicine or mathematics, the task is never ending. A knowledge base is required to be kept up to date [Davis and Lenat, 1982]. Deleting old outdated information, inserting new information and amending existing information ensures that the expert system remains an expert in its field. In some systems the refining is done solely by the knowledge engineer. First he determines where the addition, deletion or insertion is necessary and then he or she alters the program appropriately. Refining such as this is used in R1 [McDermott, 1982]. An easier way of maintaining an up-to-date knowledge base is by allowing a domain expert to interactively amend and extend the knowledge base via a special knowledge refining program [Winfield, 1982].

ELI, an expert legislative information system utilizes an interactive acquisition program. In the previous section on knowledge representation the knowledge representation of ELI (see figure 3.6) was represented with causal links connecting condition to following condition to eventual goal. Experts propose rules that have to be linked with an existing knowledge base. The details of the linking depend on how knowledge is represented. In Eli when a rule is incorporated into the knowledge base there are three main techniques which are applied to the rule in the following order. (a) Each of the input conditions of the new rule is matched against the top level conditions until a match between an input and an already assimilated condition is found. (If a match is not found then procedure (b) is followed).

Then the rule number of the input rule is attached to this matching condition and the links to lower conditions are retrieved. At this point the same matching technique is used on the conditions that are linked below the top level condition. This process will continue until there is not an existing match; then the remaining conditions from the input rule will be inserted by themselves. The goal base is then tested for a match with the input goal. If one is found then the input goal is assimilated with that goal, otherwise the input goal is appended to the existing goal list. Below figure 3.8 demonstrates the general pattern of integration.

(conditions)

(goals)

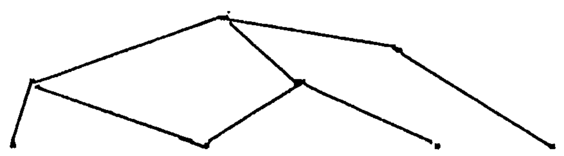


Figure 3-8: Top Down Incorporation

(b) If procedure (a) is ineffective and no match occurs with the conditions at the top level then an attempt is made to match the input goal with an already assimilated goal. If a match is not found then procedure (c) is followed. If there is a match then the process of (a) is attempted in reverse. The refining program tries to associate the input conditons with already assimilated conditions from lower levels upwards. If no further matches are found, then the remaining conditions are inserted by themselves with one being placed at the top level. Figure 3.9 illustrates the pattern of integration in procedure

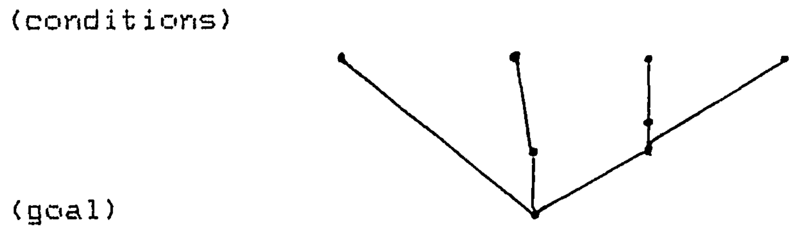


Figure 3-9: Goal Up Incorporation.

(c) IF a matching goal is not found and therefore procedure B has not been successful then the conditions are inserted as entire rules. One condition is placed at the top level, the goal is placed on the goal list, and the remaining conditons are placed in successive levels above the goal. This pattern is demonstrated in figure 3.10.

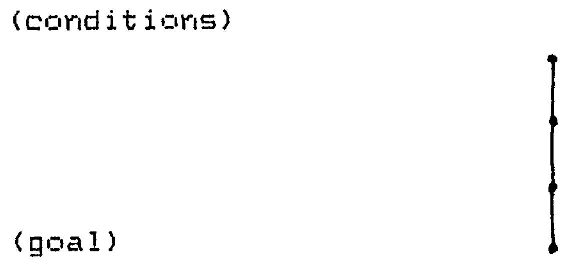


Figure 3-10: Rule Incorporated Alone

Addresses in the property list of each condition provide the links between the conditons. Rule numbers to which each conditon belongs are also an attribulte of the property list of each condition [Leith, 1983].

A more difficult approach than amending and extending the knowledge base by an interactive refining program is to allow the expert system to become self learning. It is allowing an expert system to devise its own rules from information with which it is working [Winfield, 1982].

An example of this automatic method is AM. This system exemplifies an ideal approach to accumulating knowledge. AM is given a small set of primitive facts by the system engineer and then expands those facts without further assistance from the designer. This system accumulates knowledge by positing interesting extensions to its existing concepts- either by forming new concepts or new relationships. By starting with a small number of concepts of finite set theory and a large number of heuristics about how to extend them and judge them it was able to rediscover the concept of prime numbers and the prime factorization theorem. [Davis and Lenat, 1982].

Below is an example of a heuristic used to acquire knowledge.

This heuristic proposes a new task for the AM expert

```
IF the current task (Fill-in examples of X)
   and X is a predicate,
   and more than 100 items are known in the
       domain of x,
   and at least 10 cpu seconds were spent
       trying to randomly instantiate x,
   and the ratio of successes/failures is
       both >0 and less than .05
THEN add the following task to the agenda.
   (Fill-in generalizations of x), for the
   following reason: 'x is rarely satisfied;
   a less restrictive concept might
   be more interesting.'  This reason's
   rating is computed as three times the
   ratio of nonexamples/examples found.
```

Figure 3-11: Heuristic of AM

system. When the conditions of the rule are met then this task is placed on an Agenda list of future tasks. As a result the generalizations of one concept x form new concepts in the knowledge base and AM processes each new concept to acquire the necessary information to fill in the slots of the frames involved [Lenat, 1982].

In summary, acquisition of knowledge is an important research question. By initially acquiring a large knowledge-base and thereafter maintaining it, an expert system can attain high performance. The two methods of refining an expert system are manual, interactive, and automatic. Perfecting the acquisition of knowledge automatically is a goal of expert systems in the future.

4. CENTAUR vs. R1

In this chapter two expert systems, R1 and CENTAUR are compared. The similarities and differences of their methods of representing, acquiring, utilizing, and explaining knowledge will be discussed.

As mentioned in previous chapters, R1 is a program that configures VAX-LL/780 computer systems. When provided with the input of a customer's order, it determines what, if any modifications have to be made to the order to design a functional system. As output it produces a number of diagrams manifesting how the varous components on the order are to be associated. This program is regularly used by Digital Equipment Corporations' manufacturing organization [McDermott, 1982].

The other expert system is called CENTAUR [Aikins, 1983]. It performs tasks in the domain of pulmonary (lung) physiology. CENTAUR interprets measurements from pulmonary function tests administered to patients in a pulmonary function labratory. The labratory contains equipment designed to measure the amount of gas in the lungs and the rates of flow of gases into and out of the lungs. CENTAUR is an expert consultant to the pulmonary physiologist. It produces a set of interpretation statements and a diagnosis for each patient.

4.1 knowledge representation

CENTAUR utilizes a combination of the frame and production rule methods to represent knowledge. A frame is a structure that ties together knowledge about a given situation, and provides expectations about what objects will be present in the situation and what events will occur in the situation. The frame-like structures of CENTAUR are prototypes, and prototype components. Following Minsky's frame terminology each prototype contains slots of information associated with it. Each slot provides a 'place' for information in the prototype. Missing information is therefore evident, and the system realizes how complete the solution to a problem is or is not. The system attempts during a run to fill each slot of a particular frame with a value. The value determines whether the expectations specified by the prototype are the same as those in the input.

Some of the slots in each frame are the component slots. Each compnonent is itself a frame. Therefore the value of a component slot is actually a set of 'sub-frames' of knowledge (see figure 4.1).

The frames of CENTAUR are referred to as prototypes. The prototype components contain object-level domain knowledge representing one of the principal characterizing features of the prototype. Meta-level

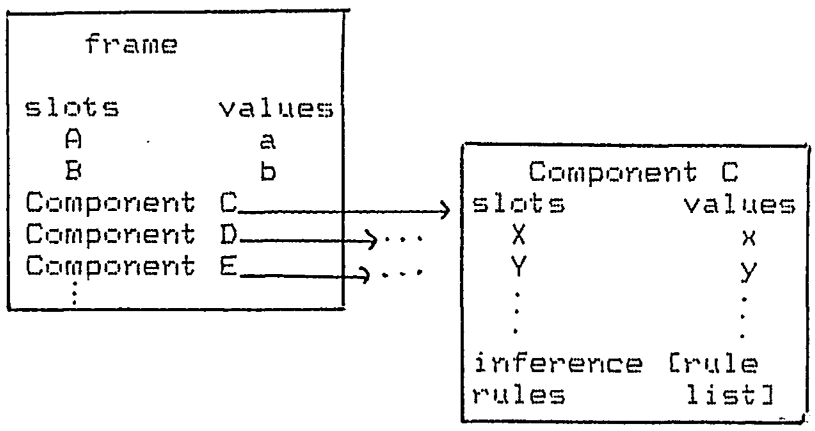


Figure 4-1: Illustration of Components

knowledge is represented by other slots in the frame. These slots include slots that control knowledge, slots that give general information about a prototype and slots specifying production rules to used during a consultation.

The frames of CENTAUR are specifically designed to complement production rules. The prototypes provide the explicit context which guides the more fine grained reasoning of the production rules. The rules are attached to slots in each prototype. Rules are one type of value for slots in a prototype. Rules are organized in a frame according to stages in which they are relevant. Each group of rules is the value of a slot representing knowledge to be applied during a particular stage of the consultation.

The CENTAUR knowledge base contains 24 prototypes, 21 of which represent disease patterns. These prototypes are linked together in a hierarchial network specifying the relationship between prototypes. A portion of the hierarchy of this group is illustrated in figure x.

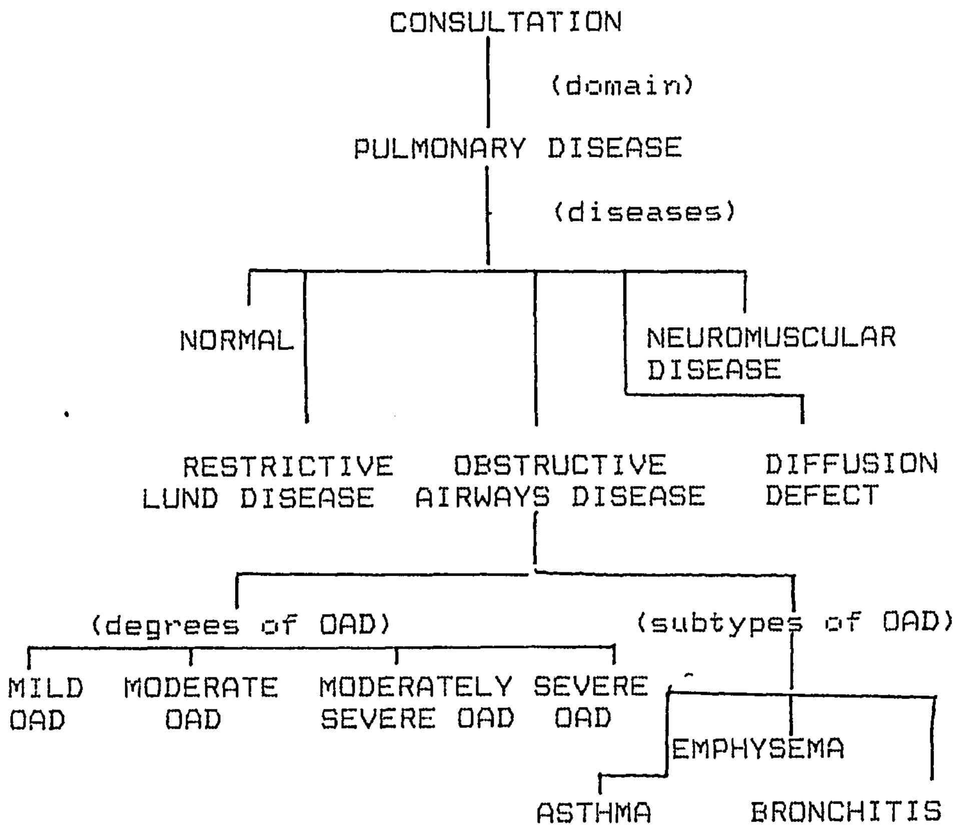


Figure 4-2: A Portion of Prototype Network.

A consultation prototype, a review prototype and pulmonary function prototype that interprets pulmonary tests comprise the remaining 3 of the 24.

The various slots of each frame are:

1. Component slots. Each prototype contains 6 to 8 component slots that point to a component frame that comprises some characteristic feature of a pulmonary disease. The component slots have a value determined by the component prototype. The component prototype evaluates the results of a pulmonary function test and reports a value representing its significance to the component slot.

2. Prototype control slots. Slots that at specific times control the consultation.

3. Prototype rule slots. There are five different types of rules. Triggering rules trigger tasks to be placed on an agenda. Inference rules are rules tried when a value is needed for a component. The remaining three rule slots are fact-residual rules that attempt to account for residual facts; refinement rules that refine diagnosis; and summary rules that summarize information.

4. General information slots include bookkeeping information and English phrases to communicate with the user.

5. Certainty measure slots indicate how certain the system is that the prototype matches the data in each case. The value of this measure ranges from -1000 to 1000.

6. Invocation records slots, such as, Intriggers and Origin slots, record information which is used in explaining why a system is exploring a given prototype.

Examples of slot values of a particular prototype are below in figure 4.3.

R1 uses a different approach. The configuration task of R1 can be viewed as a series of subtasks that have strong temporal interdependencies. Each subtask is represented in the knowledge base by production rules.

```
AUTHOR: Cohen
DATE: 9-OCT-83 17:13:29
SOURCE: Fallat
POINTERS: (degree MILD-OAD) (degree MODERATE-OAD)...
          (subtype ASTHMA) (subtype EMPHYSEMA)...
HYPOTHESIS: There is Obstructive Airways Disease.

IF-CONFIRMED: Deduce the degree of OAD
              Deduce the subtype of OAD
ACTION: Deduce any finding associated with OAD
        Print the findings associated with OAD

FACT-RESIDUAL RULES: RULE 157, RULE 158,...
REFINEMENT RULES: RULE036, RULE038, RULE039,...
SUMMARY RULES: RULE053, RULE054, RULE055, RULE083,...

COMPONENTS:
TOTAL LUNG CAPAC.    PLAUSIBLE VALUES:>100
                     IMPORTANCE MEASURE: 4
REVERSIBILITY        INFERENCE RULES: RULE
                     019,RULE020,RULE022...
                     IMPORTANCE MEASURE: 0
```

Figure 4-3: Sample Slot Values for OAD.

The first subtask is to determine whether there are major problems with the order and to rectify them if possible. This task is composed of 196 rules. The second subtask involves 87 rules for putting whatever components belong in the CPU and CPU expansion cabinet into those cabinets. The third subtask is to put boxes into the unibus expansion cabinets, and to put unibus modules into the boxes. This subtask involves 256 rules. The fourth subtask involves in its 30 rules assigning panels to cabinets and associating those panels with unibus modules and with whatever devices the modules serve. Generating a floor layout for the system is the fifth subtask of 61 rules. The last subtask is to specify what cables are to be used to connect each device to the other devices to which it has been assigned.

The rules used in the subtasks are considered domain knowledge rules. There are four types of rules involved. There are rules that generate a new subtask. Another quarter deal with adding missing prerequisite components in the order. A fourth of the domain knowledge rules create or extend partial configuration. The final fourth of rules is composed of rules that retrieve partial descriptions of components from the data base and rules that do various sorts of computations. The rules containing knowledge directly related to the subtasks add up to 480. The remaining 292 rules of R1 contain more general knowledge. Approximately one third of the remaining rules is used to generate output after the sixth subtask is completed. Another third consists of rules to exit from a subtask when there is nothing left to do. The final third of general knowledge rules is composed of rules whose function is to do counting tasks and rules that generate 'empty' data structure for the domain knowledge rules to use [McDermott, 1982].

Examples of some production rules from the sixth subtask or context are in the figure below. In expert

```
ASSIGN-POWER-SUPPLY-1
  IF: the most current active context is assigning a
          power supply
      and an SBI module of any type has been put in a
          cabinet
      and the position it occupies in the cabinet
          (its nexus)
          is known
      and there is space available in the cabinet
          for a power supply for that nexus
      and there is no available power supply
      and the voltage and frequency of the components
          on the order is known
THEN: find a power supply of that voltage and
          frequency and add it to the order.

ASSIGN-POWER-SUPPLY-2
  IF: the most current active context is assigning
          a power supply
      and an SBI module of any type has been put in
          a cabinet
      and the position it occupies in the cabinet
          (its nexus) is known
      and there is space available in the cabinet
          for a power supply for that nexus
      and there is an available power supply
THEN: put the power supply in the cabinet in the
          available space.

ASSIGN-POWER-SUPPLY-8
  IF: the most current active context is assigning
          a power supply
      and a unibus adaptor has been put in a cabinet
      and the position it occupies in the cabinet
          (its nexus) is known
      and there is space available in the cabinet for
          a power supply for that nexus
      and there is an available power supply
      and there is no H7101 regulator available
THEN: add an H7101 regulator to the order.
```

Figure 4-4: Production Rules of Sixth Subtask.

systems it is important to determine an appropriate knowledge representation in order to attain high performance. Both of these systems are successful in that aspect. CENTAUR utilizes a frame system including production rules to represent the variety of types of knowledge within its system. R1 achieves that goal by dividing its task into sequential subtasks each consisting entirely of production rules.

4.2 knowledge acquisition

How knowledge is represented in a system can be an indicator as to how well knowledge can be acquired in a system. An advantage to using production rules to represent knowledge is that they are modular. Therefore rules can be added, deleted, or modified without directly affecting other rules. They are uniform in structure with all knowledge being encoded in the same constrained syntax that can easily be understood in order to examine it or modify it.

The disadvantage of production systems is that the organization of the knowledge base makes it difficult to identify groupings of similar rules when it would be useful to make modificiations to sets of rules or in identifying interactions between rules. Adding or modifying rules can have an indirect effect on other rules when the type of explicit grouping found in various slots in a frame is not present. Furthermore, the same syntax of all rules can make it more difficult to identify the function of the knowledge of the system and therefore locate it when refining.

CENTAUR'S organization of knowledge around prototypical cases allows for knowledge acquisition. The user can easily identify the affected set of knowledge when changes to the knowledge base are desired. In CENTAUR the entire consultation process is a prototype. The various stages of the consultation are listed as separate control tasks in control slots of this prototype. This representation allows for the flexibility of adding or omitting a stage, and of more easily experimenting with the control modifications. For example, the 'refinement' stage which uses additional expertise to improve upon an interim conclusion was easily omitted during the systems early stages of development. During the consultation, points at which specific control knowledge is used are clearly defined. This results in it being less difficult than in production systems to predict the effects of the modifications that are made. Besides the consultation prototype, another prototype called review allows the user to specify one of the prototypes, and then reviews for him the 'typical' features expected in that prototype and control knowledge associated with the prototypes.

Information associated with the domain knowledge, such as the context in which the knowledge is applied, or the purpose or function of the knowledge in the consultation, is represented explicitly by each prototype. Explicit representation of control knowledge and access to reviewing frames provide a method of acquiring and refining knowledge. Another aid to knowledge aquisition is the key word CONTROL that has been defined, so that a user of the system can further inquire about the control task motivating a current line of reasoning. Furthermore, each of the component frames in CENTAUR contains a slot called inference rules. The inference rules consist of a set of production rules used to infer a value for the component. The constrained syntax of the rules also allow for ease in acquisition and modifiability of values for components.

The knowledge of CENTAUR is organized in a manner that it is easy to locate and modify the system. Several aids are an integral part of knowledge acquisition. In R1 knowledge acquisition is not quite as easy.

In R1 the major configuration task is divided into 6 subtasks. When a modification to R1's domain knowledge becomes apparent, a knowledge engineer must determine which subtask needs to be refined. In a production system, within the subtask it is not always easy to identify the function of the knowledge and therefore locate the necessary rule. However, once the offending rule is located the knowledge engineer asks the expert what he would have done differently and how he would have known to do that different thing. Sometimes in R1 a known feature of a production rule can be used to signal a different action. To make R1's performance acceptable it is only necessary to copy the offending rule and add a condition to it. Mostly, though, additional information not yet represented in R1's knowledge base is required. What McDermott refers to as 'rule splitting' in this case is necessary. One rule becomes two, the two rules discriminating between two previously undifferentiated states. However, information gathering rules for the two rules are also added to production memory [McDermott, 1982].

In summary, R1's production rules are modular. Once a rule has been located it is not difficult to modify. Centaur, using frames to represent knowledge provides a more explicit way of locating a chunk of knowledge. The prototypes represent blocks of basic knowledge that include clearly defined 'hooks' for any additional rules necessary to elaborate upon this basic knowledge. The purpose of the knowledge attached to the slots of a frame is explicit, making the effects of such modification readily predictable. However, neither system is automaticaly modifiable. CENTAUR interactively via the 'review' prototype and the 'control' function provides specific knowledge about the system. However, CENTAUR does not interactively incorporate knowledge in a fashion demonstrated by the previously mentioned ELI. Moreover, neither system appears to provide a method of modifying or acquiring knowledge for groupings of similar rules that are part of each subtask or each prototype.

4.3 knowledge utilization

To allow for the future multiple use of teaching besides diagnosing with the same knowledge base, designers of CENTAUR chose to separate the control structure within the system from the inference knowledge. Therefore, the control can later be modified without interfering with the inference knowledge.

Control knowledge in CENTAUR is represented within each prototype. This provides context specific control. The system specifies what to do in a given context as part of the domain knowledge and separates this control knowledge from inferential knowledge used in the consultation.

The control knowledge represented in prototype slots is a type of meta-knowledge applied as strategies to specify the next goal of the system. The control structure of CENTAUR can be simply stated. CENTAUR maintains an Agenda of tasks to be performed during consultation. The system interpreter executes the top task on the agenda and when the task is finished, the process repeats. When the Agenda is empty the system terminates. A task is an action to be taken by the system. It is represented as a call to a LISP predicate function. Tasks are initiated from prototype control slots and from tasks themselves as they are being executed. Each task entry includes a source for the task and a reason that a task was added to the Agenda. Tasks are executed in last-in, first-out order. Once a task is executed it is removed from the Agenda. The reasons associated with each control task are generated from the name of the prototype and the name of the control slot where the task originated (see figure 4.5). The reasons briefly explain what the system is doing.

The consultation process can be considered to proceed in stages that represent the sequence of events. Initially the system configuration for the consultation task is shown in Figure 4.6.

Knowledge in the TO-FILL-IN and IF-CONFIRMED control slots of the prototype direct these tasks.

Key stages of the consultation process including the

```
TASK:     Order the Hypothesis List.

SOURCE:   Task adding new prototype to the hypothesis
          list.

REASON:   Because new prototypes have been added
          to the Hypothesis List, it should be
          checked to see that it is ordered
          according to which prototype best
          fits the facts.
```

Figure 4-5: Task on Agenda of Centaur.

```
        AGENDA                        Current Prototype
  FILL-IN current prototype             CONSULTATION
  CONFIRM current prototype
```

Figure 4-6: Initial Configuration for Consultation.

role of the control slots are summarized below in figure 4.7.

The stages are described in more detail below:

1. Initial Data: Values for an initial set of parameters including standard pulmonary function test results are entered.

2. Triggering Prototypes: Triggering rules suggest prototypes. Certainty measures of suggested prototypes are increased.

3. Scoring and selecting a current prototype. Certainty measures determine the order of prototypes in a hypothesis list.

4. Filling in Prototype. The prototype components are filled in with facts already determined in the case. If new prototypes are

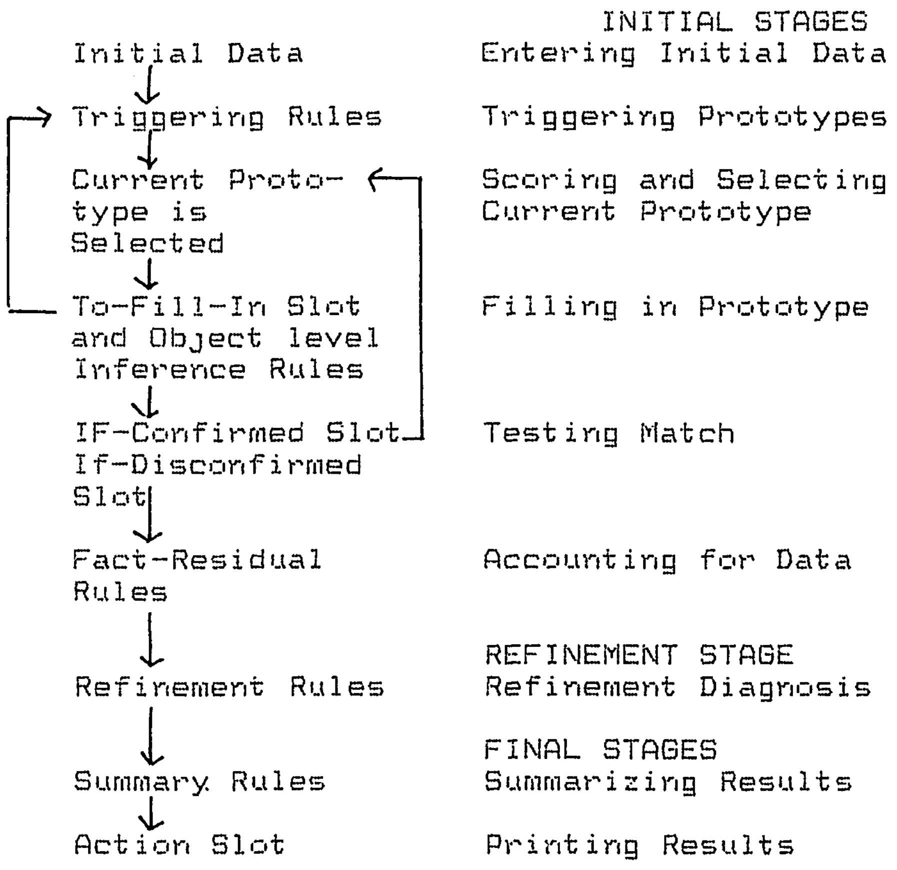


Figure 4-7: Overview of Consultation Process.

suggested then the computation returns to the 'triggering rules' stage.

5. Testing Match. An attempt is made to confirm the prototype by matching the actual facts of the case to expected values of the prototype. If tasks in the if-confirmed slot or if-disconfirmed slot suggest further sets of prototypes then the consultation returns to stage 3.

6. Accounting for Data: Fact residual rules are applied in an attempt to account for discrepancies in data.

7. Refining diagnosis: Refinement rules are applied to produce a final diagnosis of pulmonary disease.

8. Summarizing Results. Summary Rules are applied.

9. Printing Results: Tasks controlling printing are added to the agenda [Aikins, 1983].

The approach for searching for a solution in CENTAUR is called Generate and Test. In CENTAUR'S terminology prototypes represent the classes of hypotheses. One or more hypotheses are generated that explains the phenomena. These hypotheses are then tested against empirical data. Due to the hierarchy of the hypotheses only a small subset are considered at any one time. Initial input data is available to 'trigger' hypotheses classes that are most likely to match when tested. CENTAUR is unique in providing three search strategies. They are confirmation, elimination and fixed-order. The user can choose one of these three strategies to fill these slots. A 'confirmation strategy' which selects the prototype that is the best match to the data and attempts to confirm that prototype; an 'elimination strategy' which selects the prototype that is the worst match to the data and attempts to eliminate that prototype, and a 'fixed-order' strategy, which always explores prototypes in a preset order.

Each prototype selected as a possible hypothesis has a Certainty Measure, indicating how certain the system is that the prototype matches the data. The Certainty Measure ranges from -1000 to 1000. The Certainty Measure slot has a value that contains dynamic information that can change as the consultation continues. The hypothesis with the highest Certainty Measure represents the current best hypothesis. The current best hypothesis at the end of the consultation becomes the system diagnosis.

CENTAUR's control and inference methods are quite different from that of R1. The configuration task performed by R1 requires finding an acceptable configuration within a space of possible configurations. R1 always proceeds through the same sequence of subtasks. Therefore it does not require an agenda of tasks for control. R1 generates only a single hypothesis- the solution. In R1, the knowledge that other systems would use to test hypotheses is part of the generator. The inference method utilized by R1 is a form of Match. The Match method can be divided into states. Initially, Match is in a state that consists of descriptions of the components ordered for the configuration. Intermediate states are sets of descriptions of partial configurations and the as yet configured components. At each point that a decision is made, the constraint knowledge about what next step can be taken is provided by R1's rules. These is no need for backtracking in determining the next acceptable step. The final state is of course when the configuration is complete.

In the Match method, R1's rules can be divided into three categories. (1) Operator rules that take the actual next step in creating or extending a partial configuraton. (2) Sequencing rules that determine the order in which decisions need to be made so that backtracking is not necessary. (3) Information gathering rules provide the information needed for operator and sequencing rule selection. The consequences of applying an operator must bear only on aspects of the solution that have not yet been determined.

Match, however is not capable of performing the entire inference task. The subtask of placing modules in the unibus is formulated by a Generate and Test method that finds an optimal sequence that fits within spatial and power-load constraints.

In summary, the use of Match as an inference method is appropriate to the structure of the configuration domain. It avoids search and limits the cost of running the program. An Agenda for control and the more typical inference method of Generate and Test is used in CENTAUR and is more appropriate for its analytical task. CENTAUR is given a single complex set of data and the task of the program is to decompose the data and determine the relationship of the data. The configuration task of R1 is synthetic. R1 is given a set of components and its task is to impose relationships on those components and form a complex object.

The explanation capabilities of a system are a critical factor in the acceptance by users of large knowledge-based consultation systems. Both CENTAUR and R1 have explanation capabilities referred to as tracing. In CENTAUR tracing can be performed during the consultation at different levels that range from 0 to 3. The user is asked in the initial stage of the consultation what level of trace he requires. The explanation of how the system is coming to a particular conclusion is then placed in brackets throughout the consultation. An example of tracing is illustrated in figure 4.8 below.

The trace [Trigger for ASTHMA and CM 900] explains to the user that his response to the question referral diagnosis has triggered the system to generate the hypothesis that asthma is the diagnosis with a certainty measure of 900. The trigger for the Normal prototype refers to no disease in the patient. OAD (obstructive airways disease) is another hypothesis that the system

```
Tracing level (0-3)
**2
Agenda Printing?
**No
Consultation Strategy:
**Confirmation

(The two stars preceeding a comment represent the
users response.  The user has chosen a tracing
level of 2.  Below information given as
explanations of the trace are within brackets.)

-------PATIENT -7----------
1)Patient's identifying number:
**7446
2)referral diagnosis:
**ASTHMA
  [Trigger for ASTHMA and CM 900]
3)RV/RV-predicted:
**261
4)TLC (body box) observed/predicted:
**139
5)FVC/FVC-predicted:
**81
  [Trigger for NORMAL and CM 500]
6)FEV1/FVC ratio:
**40
  [Trigger for OAD and CM 900]
```

Figure 4-8: Illustration of explanation facility.

explains is being considered.

In R1 it appears that there is one level of tracing in the system. The trace is separate from the output of the system and describes the process and subtasks followed to attain a configuration (see figure 4.9) of the order.

i. MAJOR-SUBTASK-TRANSITION

```
2.      SET-UP
3.            UNBUNDLE-COMPONENTS
53.            NOTE-CUSTOMER-GENERATED-EXCEPTIONS
55.            NOTE-UNSUPPORTED-COMPONENTS
57.             CHECK-VOLTAGE-AND-FREQUENCY
104.            CHECK-FOR-TYPE-OR-CLASS-CHANGES
110.            VERIFY-SBI-AND-MS-DEVICE-ADEQUACY
111.                COUNT-SBI-MODULES-AND-MB-DEVICES
126.                    GET-NUMBER-OF-BYTES-AND-COUNT-
                           CONTROLLERS
137.            FIND-UBA-MBA-CAPACITY-AND-USE
145.             VERIFY-MEMORY-ADEQUACY
146.                PARTITION-MEMORY
160.        ASSING-UB-MODULES-EXCEPT-THOSE-CONNECTING-
               TO-PANELS
177.        VERIFY-UB-MODULES-FOR-DEVICES-CONNECTING-
               TO-PANELS
                FIND-ATTRIBUTE-OF-TYPE-IN-SYSTEM
179.                VERIFY-COMPONENT-OF-SYSTEM
207.        NOTE-POSSIBLY-FORGOTTEN-CONPONENTS
213.        CHECK-FOR-MISSING-ESSENTIAL-COMPONENTS
215.MAJOR-SUBTASK-TRANSITION
216.     DELETE-UNNEEDED-ELEMENTS-FROM-WM....
```

Within both systems it appears that explanatory knowledge is not represented separately from its performance knowledge. It, therefore, does not need to be modified when changes are made to the performance knowledge. The method of tracing in CENTAUR explains as the program is arriving at decisions the reason for these decisions. An expert user is better able to understand the process of the consultation program than in R1. R1 describes the trace separately from the output.

In summary, there are many differences and few similarities between the expert systems CENTAUR and R1. CENTAUR utilizes basically a frame method for knowledge representation. Although, the individual frames do include production rules. R1's knowledge, in contrast is entirely represented by production rules.

CENTAUR's system has several tools for knowledge acquisition. The systems explicit representation of control knowledge, the 'review' prototype and the CONTROL function provide aids in locating where modificiations are necessary. The constrained syntax of the inference rules in the system makes modifications easy. Because R1 is not a frame system, locating the rules to be changed is slightly more difficult. However, the six sequential subtasks of R1 allow the user to determine fairly easily which subtask is involved. Once the rules for modification are located, R1 either adds conditions to the particular rule or splits the rule into two rules- thus developing each rule separately.

Another contrast between R1 and CENTAUR is their method of utilizing knowledge. CENTAUR's control knowledge is located within the individual frames and is separate from inference knowledge of the system. The control knowledge provides information to an Agenda of tasks that executes tasks on a last-in, first-out basis. The inference method is 'generate and test'. Hypotheses are triggered from initial input data and then tested to confirm if they are the best diagnosis. Searching of hypotheses is done by confirmation, elimination, or fixed-order strategies. R1's system is different. The subtasks provide a structure that eliminates the need for an Agenda. The inference method is called 'Match'. The first stage of 'Match' consists of descriptions of components ordered. Intermediate stages contain sets of descriptions of partial configurations and yet unfigured components. The final stage is the total configuration.

Both R1 and CENTAUR use trace to provide the user with explanations. The trace in CENTAUR is more elaborate and is available during the consultation.

Although the systems are very differnt they are each successful in providing expert information to the user. The methods of representation, acquisition, utilization and explanation are appropriate to their individual requirements.

References

Heuristic Programming Project 1980. Heuristic Programming Project. Stanford: Stanford University 1980.

Addis, T.R. Knowledge Refining for a Diagnostic Aid (An Example of Applied Epistemics). J. Man-Mach. Stud., 1982, 17, 151-64.

Aikins, J. Prototypical Knowledge for Expert Systems. Artif. Intelligence, 1983, 20, 163-210.

Barr, A.; Feigenbaum, E. A. The Handbook of Artificial Intelligence. Stanford: Computer Science Department, Stanford University 1980.

Bendifallah, S. Knowledge-based Decision Support Systems: Social Significance, pages 292-98. Society for General Systems Research with the American Association for the Advancement of Science, 1982.

Bleich, H. L. Computer Evaluation of Acid-base Disorders. J. Clin. Invest., 1969, 48, 1689-1696.

Bobrow, D. G. Representation and Understanding. New York: Academic Press 1975.

Bonnet, A.; Cordier, M. O.; Kayser, D. An ICAI System for Teaching Derivatives in Mathematics, pages 135-41. IFIP World Conference on Computers in Education, 1981.

Boothroyd, D. The Revolution of the Non-Expert System. Engineer , 1982, 255, 59-63.

Burkitt, A. How Donald Michie Was Flipped Off His Feet By AI (Artificial Intelligence). Computing, 1982, 10, 32-33.

Burton, R. R. Semantic Grammar: an Engineering Technique for Constructing Natural Language Understanding Systems. : Bolt Beranek and Newman 1976.

Catanzarite, V. A.; Greenburg, A. G.; Bremermann, H. J. Computer Consultation in Neurology: A Review of the Field, and a Comparison of the Neurologist System to Previous Programs. J. Clin. Comput., 1981, 10, 64-84.

Chester, D. Elements of Knowledge-Based Expert Systems, pages 42-48. , 1982.

Couch, R. D. Computer Diagnosis Review and Comment. Path. Ann., 1976, 11, 141-159.

Davis, R.; King, J. An Overview of Production Systems. Machine Intelligence, 1977, 8, 300-322.

Davis, R.; Lenat, D. Knowledge Based Systems in Artificial Intelligence. New York: McGraw-Hill 1982.

Ellis, P. Expert Systems- A Key Innovation in Professional and Managerial Problem Solving. Inf. Age , 1983, 5, 2-6.

Erman, L. D.; Hayes-Roth, F.; Lesser, V. R.; Reddy, D. R.; The HEARSAY-II Speech Understanding System Integrating Knowledge to Resolve Uncertainty. ACM Comput. Surveys, 1980, 12, 213-253.

Fagan, J. M. VM: Representing time-dependent Relations in a Medical Setting. PhD thesis, Stanford University, 1980.

Fagan, L. M.; Kunz, J. C.; Feigenbaum, E. A.; Osborn, J. J. Representation of Dynamic Clinical Knowledge Measurement Interpretation in the Intensive Care Unit, pages . Internat. Joint Cionf. Artificial Intelligence, 1979.

Feigenbaum, E. A. The Art of Artificial Intelligence: I. Themes and Case Studies of Knowledge Engineering, pages 1014-1029. Internat. Joint Conf. Artif. Intelligence, 1977.

Feigenbaum, E. A. Innovation and Symbol Manipulation in Fifth Generation Computer Systems, pages 223-6. , 1982.

Goodall, A. Language of Intelligence (Prolog). Syst. Int. , 1983, 11, 21-4.

Gorry, G. A. Knowledge-based Systems for Clinical Problem Solving. Decision Making and Medical Care, 1976, , 23.

Gowans, J. Expert Systems. Microcomput. Printout , 1982, 3, 66-67.

Innocent, P.R. Towards Self-Adaptive Interface Systems. Int. J. Man-Mach. Stud. , 1982, 16, 287-99.

Kaihara, S.; Koyama, T. Medical Consultation System with Practical Requirements-Development of Mecs-AI. In Computer Science and Technologies, : North-Holland, 1982.

Leith, P. Hierarchically Structured Production Rules. Comput. J. , 1983, 26, 1-5.

Lenat, D. B. The Nature of Heuristics. Artif. Intell. , 1982, 19, 189-249.

Lindsay, R. K.; Buchanan, B. G.; Feigenbaum, E. A.; Lederberg, J. Applications of Artificial Intelligence for Organic Chemistry- the DENDRAL Project. New York: McGraw-Hill 1980.

Martin, W. A. and Fateman, R. J. The MACSYMA System, pages 23-25. Symposium on Symbolic and Algebraic Manipulation, 1971.

McCartney, J. Expert Systems: Expertise Captured in a System. Data Processing, 1982, 24, 26-27.

McCarty, L. T.; Sridharan, N.S.; Sangster, B.C. The Implementation of TAXMAN II: An Experiment in Artificial Intelligence and Legal Reasoning. Technical Report, Rutgers University, 1979.

McDermott, J. R1: an Expert in the Computer Systems Domain, pages 269-271. Conference of Artificial Intelligence, 1980.

McDermott, J. R1: A Rule-based Configurer of Computer Systems. Artif. Intell. , 1982, 19, 39-88.

Meadow, C.T.; Hewett, T. T.; Aversa, E.S. A Computer Intermediary for Interactive Database Searching, I. Design. J. Am. Soc. Inf. Sci., 1982, 33, 325-332.

Minsky, M. A Framework for Representing Knowledge. AI Memo 306.

Mizoguchi, F. A Software Environment for Developing Knowledge Base Systems. In Computer Science and

Technologies, : North-Holland, 1982.

Newell, A. Some Problems of Basic Organization in Problem- Solving Programs, pages . , 1962.

Parrott, J. Why Artificial Challenges Need Real Answers. Comput. Manage. , 1983, , 18-20.

Pinkerton, J. M. M. The Advance of Information Technology. ICL. Tech. J., 1982, 3, 119-36.

Rundle, T. The Getting of Knowledge (Expert Systems). Comput. Syst., 1982, 2, 41-42.

Sacerdoti, E. D. Planning in a Hierarchy of Abstraction Spaces. Artif. Intelligence, 1974, 5, 115-135.

Smith, M. F.; Bowen, J. A. Knowledge and Experience-based Systems for Analysis and Design of Microprocessor Applications Hardware. Microprocess. and Microsyst. , 1982, 5, 515-18.

Stefik, M. J. Inferring DNA Structures from Segmentation Data. Artif. Intelligence, 1978, , 85-114.

Stefik, M. J. Planning with Constraints. Artif. Intelligence, 1981, 16, 111-140.

Stefik, M.; Aikins, J.; Balzer, R.; Bendit, J.; Birnbaum, L; Hayes-Roth, F.; Sacerdoti, E. The Organization of Expert Systems, A Tutorial. Artif. Intell, 1982, 18, 135-173.

Sumner, G. C. Knowledge-based Systems Maintenance Applications (ATE), pages 472-473. , 1982.

Sussman, G. J.; Steel, G. L. CONSTRAINTS- A Language for Expressing Almost-hierarchial Descriptions. Artificial Intelligence, 1980, , 1-39.

Tanaka, K. Resume of Knowledge Engineering in Japan. In Computer Science ad Technologies, : North-Holland, 1982.

Wallis, J. W.; Shortliffe, E.H. Explanatory Power for Medical Expert Systems; Studies in the Representation of Causal Relationships for Clinical Consultations. Methods Inf. Med., 1982, 21, 127-136.

Waterman, D. A. A Rule-based Approach to Knowledge Acquisition for Man-machine Interaction. _International Journal of Man-Machine Studies_, 1978, _10_, 693-711.

Webster, R. Planting an Expert. _Micro_. _Decis_. , 1982, _14_, 107-8.

Winfield, M. J. Expert Systems: An Introduction for the Layman. _Comput_. _Bull_. , 1982, _2_, 6-7.

Winston, P. _Artificial Intelligence_. Reading: Addison-Wesley Publishing, Inc. 1977.

Zadeh, L. A. A Theory of Approximate Reasoning. _Machine Intelligence_, 1979, , .

Vita

The author was born in New York City on March 24, 1950 to Ned and Freda Greenberg. She became a Rotary Exchange Student in Argentina from 1966-1967.

Mrs. Cohen graduated from Pennsylvania State University with High Distinction in 1971 with a Bachelors of Science. While there Mrs. Cohen became a menber of Alpha Lamda Delta freshman sorority; CWENS national sophomore society for academic and extra curricular activities; and Phi Kappa Phi national honor society for high academic performance in the sciences. In 1978, she received a Master of Education degree from Lehigh University.

During her professional career she has taught elementary school. Mrs. Cohen then became a Reading Specialist at the elementary and college levels. She became a graduate student in Computer Science in January, 1982 at Lehigh University. In the summer of 1982 she was a graduate assistant in that field.

Mrs. Cohen is married to Robert Cohen and has two children, Zachary and Samuel.