NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

EXPERT SYSTEMS FOR MEDICAL DIAGNOSIS

Graduation Project
COM-400

Student : Bahri YİĞİT (950377)

Supervisor : Asst. Prof .Dr .Rahib ABIYEV

Lefkosa-2002

ACKNOWLEDGEMENT

First of all I would like to thank my Graduation Project Supervisor Assoc.Prof.Dr. Rahib Abiyev who is a patient & very appreciating personality .He has guided me with a keen interest and helped me by all means. Dr.Rahib, thanks for your continual support.

I would like to thank all my teachers in the Near East University, including Faculty of Engineering Dean Prof.Dr. Fakhreddin Mamedov, Department of computer Engineering Chairman Assist.Prof.Dr. Adnan Khasman, Student Advisors Miss. Besime Erin and Mr. Tayseer Alshanableh and all the Staffofthe Faculty of Engineering, and special thanks to the Vice-President of Near East University Assoc.Prof.Dr. Şenol Bektaş for giving me the opportunity to experience this remarkable Institute that have showed me preliminary steps towards my professional carrier.

Many thanks to my parents & sisters for their continuous moral support and encouragement.

ABSTRACT

Increasing the complexity of the technology processes, the presence of difficult formalization and unpredictable information, the uncertainty of environment Leads to non-adequate description of these processes by deterministic methods, and so the development of control system with low accuracy. The effective way to solve this problem is the use of artificial intelligence ideas, such as expert systems.

The aim of this thesis is the development of expert system for medical diagnostics. For this purpose the state of art understanding of expert system for diagnostics problem solving is given, the structure of expert system and the functions of its main blocks are described.

Models of knowledge representation, such as OAV triplets, semantic networks, predicate, logics, frames, neural networks, rule-based model is chosen, their main properties are widely described. After the analysis of knowledge acquisition and their realization are considered. As an example, the development of diagnostics expert system for stomach and intestine diseases is considered. Using experienced expert knowledge and different medical references the knowledge-based is created. This knowledge-based has about 256 production rules. Premise part of the rules includes the input features of stomach diseases, and the conclusion part includes diagnosis. The considered expert system is realized on the base of ESPLAN expert system shell.

TABLE OF CONTENTS

1. APPLICATION OF EXPERT SYSTEM FOR SOLVING DIAGNOSTIC PROBLEMS

1.1. The Expert System Concept

The expert system is a recent addition to circle information systems. Expert systems are computer-based systems that help managers resolve problems or make better decisions. However, expert systems, which are also referred to case-based reasoning systems, do so with decidedly different twist. An expert system is an interactive computer based- system that responds to questions, asks for clarification, makes recommendations, and generally helps the user in the decision-making process. In effect, working with an expert system is much like working directly with human expert to solve a problem. It even uses information supplied by a real expert system in a particular field such as medicine, taxes, or geology. Expert systems re-create the decision process better than humans do. We tend to miss important considerations or alterative computers don 't.

An expert system applies preset IF- THEN rules to solve a particular problem, such as determining a patient's illness. Like management information systems and decision support systems, expert systems rely on factual knowledge, but expert systems also rely heuristic knowledge and the heuristic rules of thumb used in an expert system are acquired from a relative domain expert system, a human expert in a particular field, such as jet engine repair, life insurance, or property assessment. The expert system uses this human-supplied knowledge the human thought process within a particular area of expertise. Once completed, an expert system can approximate the Logic of a well-informed human decision-maker.

An expert system is a computer program that represents and reasons with knowledge o special subject with a view to solving problems of giving advice.

An expert system may completely fulfil a function that normally requires human expertise, or it may play the role an assistant to human decision-maker. In order words, the client may interact with the program directly, or interact with human expert who interacts with the program. The decision-maker may be expert in his own right, in which case the program may justify its existence by improving his productivity.

Expert system technology derives from the search discipline of Artificial Intelligence (AI): a branch of COMPUTER Science consumed with a design and implementation of

programs, which are capable of emulating human cognitive skills such as problem solving, visual perception and Language understanding. The typical tasks for expert systems involve:

- The interpretation of data
- Diagnosis of malfunctions
- Structural analysis of complex objects
- Configuration of complex objects
- Planning sequences of actions

1.2. The Characteristics of an Expert System

The most obvious feature of an expert system is that it operates as an interactive system that responds to questions, asks for clarifications, makes recommendations and generally aids the decision-making process. To a user, this interactive interface is what would distinguish an expert system from any ordinary computer tool. Behind this interface Lie other characteristics that may not be immediately obvious to a person using the tool.

Expert system tools have the ability to store and sift through significant amounts of knowledge. There are various mechanisms used in the storage and retrieval of knowledge, some of which shall be discussed in the next section. An expert system needs a large knowledge base in order to be able to tackle any kind of problem that may arise within its area of expertise.

Not only must such a system be able to store the available knowledge, but it must also support mechanisms to expand and improve the knowledge base on a continuing basis. Every specialized field is always in a state of flux, with something new being discovered all the time. In order to keep the expert system up-to-date, it is necessary to leave the knowledge base open-ended so that new pieces of information can be added at any time, without need for significant changes in the structure of the system.

An expert system must have the capability to make Logical inferences based on the knowledge stored. This is where the simple reasoning mechanisms used in expert systems come into play. This is what makes an expert system tick. A knowledge base, without any means of exploiting the knowledge stored, is useless. This would be analogous to Learning all the words in a new Language, without knowing how to combine those words to form a meaningful sentence.

A feature somewhat unique to expert systems is that a particular system caters to a relatively narrow area of specialization. Expert systems are very domain-specific. A medical expert system cannot be used to find faults in the design of an electrical circuit. This focus on small domains is more a result of technological limitations than anything else. As discussed earlier, the quality of advice offered by an expert system is dependent on the amount of knowledge stored. As the scope of an expert system is widened, its knowledge base needs to be expanded. The methodologies available today limit the amount of knowledge that can be stored and retrieved in reasonable amounts of time.

1.3. Decision Making

Decision-making ranges from the routine and swift to the complex and consuming. Decision-making implies the existence of a minimum of the following four factors:

1. There must be a problem.
2. There must be a decision-maker.
3. There must be alternative solutions to the problem,

Given that these four elements do exist, there are a variety of methods through which one may derive candidate solutions to the problem under consideration -for presentation to the decision-maker. The discipline devoted to the development and implementation of such tools may be called decision analysis. Those who work within this discipline and who ultimately present the alterative solutions to the decision-maker are called decision analysts.

To better understand expert systems, it is vital to understand and appreciate decision analysis, its supporting elements, and its role in the decision making process. in particular, it is anticipated that through such decision, one may more fully appreciate just when and where to employ expert systems.

Obviously, decision-making is hard new concept. Human beings have been making decisions ever since human Life first appeared on this planet. Cave dwellers had to decide where to Live, what to hunt, when to hunt. In making these decisions it is extremely doubtful that they are any rigorous approach to assist them substantiating or improving those decisions made. Intuition, experience, and judgment reached those decisions strictly.

In more recent times, and in particular and the past few centuries humans have developed, and have begun to reply on, more formal and rigorous means for assistance

In their decision making. Such means have been achieved through an increased dependence on the use of decision models, and r particularly on quantitative models and analytical methods. Today their reliance of corporations and institutions on such techniques as follows:

- Spreadsheets and databases
- Statistical analysis
- Simulation
- Methods of mathematical optimisation

While this formal approach to decision-making has certainly not sub planted the use of intuition, experience, and judgment, it has found acceptance and use as an adjunct to the decision making process. Typically, when we utilize this more explicit, analytically base approach to decision support, we call it decision analysis to distinguish .It from the qualities aspects involved in making decisions. However, ultimately both qualities and quantitative factors must be taken into account in the decision making process.

The purpose of decision analysis is to provide the decision-maker with information for use in the support of the decision making process, where such information has been derived through a logical and systematic process.

1.4. DP, MIS & DSS

One way in which decision analysis might reasonably be viewed for a process that involves transformation of the data into (useful) information support of the decision making process. As our civilizations have evolved, we have become great collectors of data. Unfortunately, data alone are of Little benefit. To have value, data must be transformed into a format from which we can perceive such useful information trends, measure of central tendency, and measures of dispersion or variability.

One fundamental rule data is that, to be value, data must be in the right form, in the right place, at the right time.

1.4.1 Data Processing (DP)

The simplest method for the transformation of data is that of data processing, or DP. Typically, the DP approach is used to transform a set of raw data into the following information:

- Statistics
- Pictorial representations

For example, consider a problem in which data have been collected on engine failure for the specific type of military aircraft at several different bases. To simplify our decisions, assume that each base has the same number of total aircraft and each flies the same number of missions each month. Twelve months of data are given in table.

The data in table are termed raw data as they simple in the form in which they were originally collected. We may also consider these data to be our engine data failure database.

Table: Engine Failure Data

Month	Base Failures A	Base Failures B	Base Failures C
1	5	3	6
2	1	2	7
3	4	2	5
4	3	1	2
5	7	0	2
6	4	2	3
7	1	2	2
8	7	2	2
9	4	3	3
10	6	1	5
11	6	1	7
12	5	2	7

Now, even though our data processing has been elementary and incomplete, we should still find it easier to make the following observations:

- The average monthly number of engine at bases A and C are more than twice those of base B.
- A trend in engine failures at base C seems possible. That failure appears to increase in the winter months and decrease in the summer.

Thus, from this simple illustration, we can see the usefulness of even a very

rudimentary level of data processing.

1.4.2 Management Information Systems (MIS)

The next Level sophistication in the processing of data information is called management information systems, or MIS. While there is no uniform agreement about the precise definition of MIS, the general intent of the earliest such system was to provide information directly, and in real time, to the decision-makers. And in a format compatible with their style and needs for decision-making. Information is the bases upon which managers may purpose their duties, specifically the duties of planning, organizing, staffing. And control. MIS certainly existed before the advent of the computer; it is now customary to think of a MIS as a system that finishes management information by means of a digital computer and connecting information network. The typical MIS concept involves a computer console display at the decision-maker's desk.

1.4.3 Decision Support Systems (DSS)

As may be noted from the above discussion, MIS are relatively passive entities. While they remove mush of the drudgery of the data processing and the development of visual aids, and substantially decrease the time required to obtain such information, they still play a Limited role in decision-making. However, at about the same time that MIS was becoming popular, developments were taking place in other fields that addressed the implementation of certain analytical methods for decision analysis. In particular, representative mathematical models of certain classes of problems were being formulated and various methods for providing solutions to the models were constructed. Including among such methods are following:

- Mathematical programming
- Marginal analysis
- Input-output analysis
- Queuing theory
- Inventory theory
- Project scheduling
- Simulation
- Reliability and quality control
- Forecasting
- Group technology
- Material requirements planning

Assuming that can represent our specific problem using one or more of such models, the associated methodology may then be used to develop a proposed solution. However, as the critics of such approaches have noted, to accomplish this, one is required to transform a real world problem into a mathematical model.

While some advocates of DSS 's might disagree one may think of a DSS as a combination of a MIS and the analytical tools as Listed above. Thus one conception of a DSS is that computerized system for accessing and processing data, development managerial displays, and providing recommended courses of action as developed through the use modem analytical methods. Using this definition, a block diagram for a general DSS is depicted in Figure at below:

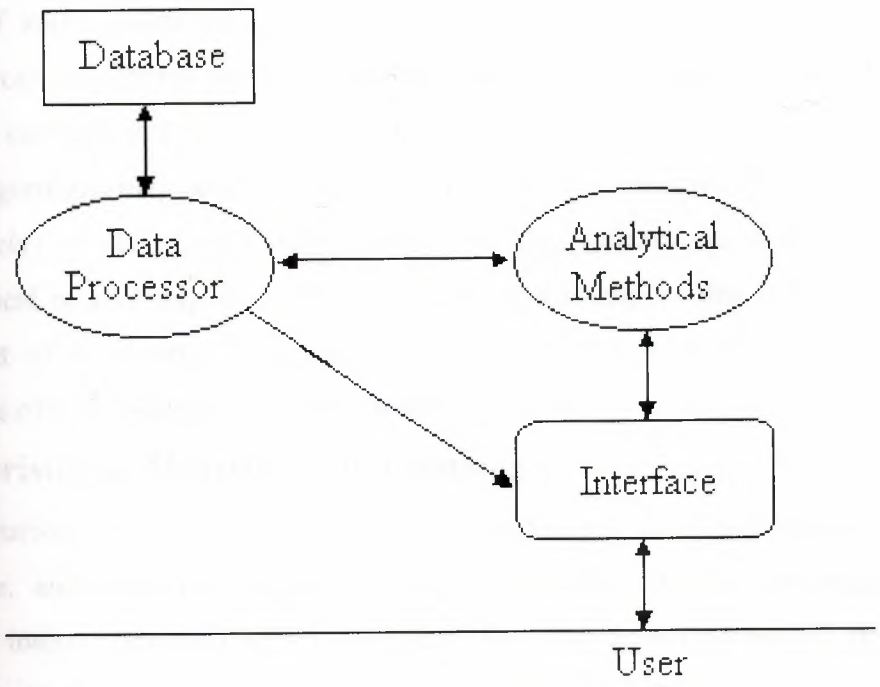


Figure 1.1: A Genetic DDS.

As in the case of the MIS, the DSS would access the database and develop displays in the appropriate format. However, assuming that our DSS includes a

Supporting tool for the solution of scheduling problems, the manager will also be provided with a recommended schedule for production as generated by the scheduling methodology.

The manager May then either accept the DSS recommendation or develop his or her own schedule-which may be compared with one developed by the DSS through a simulation of the proposed schedule, for example. Thus, a DSS is certainly a far more active participant in the decision-making procedure than either DP or MIS.

Through discussion of DSS, we have referred rather casually to analytical methods. Such methods normally invoke the use of algorithms for the derivation of the solutions for the particular class of mathematical model under consideration. T o more fully appreciate the DSS concept, as well as the difference between DSS and expert systems, we need to understand algorithms. .

1.5. Algorithms and Relationships

One formal definition of an algorithm is, a method for solving a problem using operations from given set of basic operations which produces the answer. In a finite number of such operations. Typically, these basic operations are simply elementary mathematical procedures such as addition, subtraction, multiplication, and division. Note most carefully that this definition implies that an algorithm converges.

Algorithms may be applied to an either single mathematical relationship or (and more Likely) to set of such relationships, for the purpose deriving a solution. A mathematical relationship is simply a mathematical statement that relates the various component of a system. In other word, a relationship is a representation of out knowledge of how a particular systems works.

1.6. Heuristics & Heuristic Programming

Heuristic rules, or heuristics for short, are that are developed through intuition, experience, and judgment. Typically, they do not represent our knowledge of the design, or interrelationships within, a system. Heuristics do not necessarily result in the best, or optimal, result. Heuristics are often called rules of thumb. For example, consider the following heuristics:

- Don't ask the boss for a raise if he is in bad moon.
- Avoid Houston's Southwest freeway during the rush our.
- Sell a stock if the dividends are to be cut.
- Buy gold an inflation hedge.

One of the general characteristics of many heuristics is their focus on screening, filtering and pruning. Each of these terms represents just another way to sate that heuristics may be used to reduce the number alternatives that are considered. Typically

an expert learns through time and experience that certain approaches tend to work well, while others do not.

When one or more heuristics are combined with a procedure for deriving a solution from these rules, we have a heuristic program. Ass in the case of algorithms, heuristic programming involves finding a solution to a problem using operations from a given set of basic operations, where such a solution is produced in a finite number of such operations. However, and this is the main difference between algorithmic produces and heuristic programming, the solution found may or may not be a theoretically best possible answer.

Not that when one uses heuristics, heuristic programming, and one is implicitly accepting the notation satisfying. Satisfying is concept for use in the explanation of how individuals and organizations actually arrive at decisions. Specifically, we typically do not seek optimal solution; rather we seek an acceptable solution. Heuristics (heuristic programs) are then indented for use in obtaining acceptable solutions. However, we can only justify the use of heuristics in those cases for which more formal analytical methods (in particular, methods that develop optimal solutions) would prove less effective.

At some point, even such mathematically sophisticated approaches will no longer work. This is, because such a problem exhibits has been called combinatorial explosiveness. That is, the time required solving such problems increases exponentially with problem size. In such instance, we might be well advised to heuristics and heuristic programming simply because of the computational complexity of the problem.

In heuristic programming, we have, in essence, the same situation. That is the heuristic rules coming with the steps of the solution procedure. However, in this instance, our solution procedure is not algorithm, as it is does not guarantee an optimal solution. On designation for the solution procedure is that of an inference process -the procedure which serves to infer conclusions from the set of the heuristic programming for processing on a machine. The heuristics used to provide a solution (schedule) for this problem involve the following:

- Schedule jobs with shorter processing times before those with Longer ones.
- If two (or more) jobs are tied for processing trines, give priority to the job that is mostly tardy -or Likely to become tardy.

Application of these rules, through a heuristic program, will certainly result in a schedule. Hopefully, such a schedule might even be a good one -but there are no

guarantees as to how close, or far, we might be from the optimal schedule. In order to keep the discussion simple, we may do conclude that heuristics and heuristic programming, when and where appropriate may enhance one's decision-making procedure. As such, these methods often form a portion of the tools incorporated into decision support systems and serve to alleviate the limitations of the more rigorous analytical techniques.

1.7. Artificial Intelligence

Artificial intelligence, or AI consumed with precisely the same problem that DSS a heuristic programming are concerned with, that is, decision making. One fundamental difference is the objective of those in the AI community considerably more ambitious than that of the DSS sector. The purpose of AI is not simply to support decision- making, or making to enhance decision-making; rather, the ultimate goal of AI is to. Develop an intelligent machine that will itself make decisions. In particular, this intelligent machine should exhibit intelligence on the same order as that a human.

An intriguing definition of AI is, AI is the study of how the make computers do things at which, at the moment, problem are better". Using this definition, we may avoid the problems of either the definition of determination of the existence of intelligence and instead, simply compare the computer's performance (in some are) with that of humans.

1.8. Certain Differences of Opinion

Much of the criticism now being directed toward expert systems is, we believe, due to rush become involved with the methodology coupled with a failure to take the time and effort to truly understand and appreciate the concept, its history, scope, and Imitations. In addition, we must admit to disagreement with a number of commonly held perceptions of, and practices within, the expert systems. The include, in particular, the following:

- The implication in the Literature, through omission of statements to the contrary, those expert systems can and should be used in virtually any problem And failure to emphasize the importance of having a reasonable familiarity with the alternative solution procedures.
- The implication that, to understand and use expert systems, you must be familiar with certain AI Languages (LISP and PROLOG)

- Statements that imply that expert systems is just an alterative and conventional computer programming.
- The emphasis, in too much of the expert systems Literature, on those factors that really only support expert systems.
- The widely held belief that knowledge engineer is synonymous with a computer programmer -or computer scientist.
- The implication that one way learn how to use expert systems by simply learning how to run a commercial expert systems software package -and the resulting the development of expert systems software technicians, rather than competent knowledge engineers.
- The belief that, just because a person doing a job, he or she is an expert in that job -ant the concomitant cloning mediocrity.
- The widespread belief that best, if not only to validate the performance of an expert system is to compare its performance.
- The belief that potential expert systems developers should look at applications that have the potential of either saving the company or earning for the company several million dollars a year. This further implies that the only expert systems worth building are those involving many hundreds or thousands of rules.

1.9. Application of Expert Systems

1.9.1. DENDRAL-An Expert in Chemical Identification

Work on DENDRAL; generally considered to be the very first Expert System. The purpose of DENDRAL, which did not actually become operational until the early 1970s is the identification of the molecular structure of unknown compounds, a problem of considerable computational complexity. DENDRAL, unlike many of the early expert systems found acceptance and is still in use by chemists all over the world. The purpose of the collaboration was to determine if heuristics could be used to develop results comparable to the algorithm, but in less time. DENDRAL utilizes production rules and was implemented in the LISP programming language.

1.9.2. HEARSA Y 1 & 11- Speech Recognition

HEARSA Y l & ll were developed in attempt demonstrate the possibility of a speech recognition system. Specifically, the goal of the system was to have a computer understand spoken input. The input to the HEARSA y system is a speech waveform.

From this waveform, a set of hypotheses about may have been said is developed. A best guess from this set is then presented as the output.

One of the more innovative concepts developed by the HEARSA Y project was that of the use of multiple knowledge bases.

At the completion of the HEARSA Y project in 1975, the system had a vocabulary of about 1000 words and was able to correctly interpret spoken input roughly 75 percent of the time.

One of the important results of this project was the demonstration that an expert systems approach was superior to what had been the conventional approach to speech recognition. Included among these are HASP/SIAP systems.

1.9.3. INTERNISTICANDUCEUS -An Expert in Internal Medicine

The INTERNIST project was started in the early 1970s, and continues today under CADUCEUS. One of the truly striking things about. INTERNISTICANDUCEUS has been its ability to remain viable project over such an extensive period time.

The goal of INTERNIST is to perform diagnosis of the majority of diseases associated with the field of internal medicine. This, in itself, is an ambitious endeavour as there are hundreds of such diseases.

1.9.4. MYCIN -An Expert in Blood Infections

MYCIN is, at this time, probably the most widely known of all expert systems. Despite this fact that it has never been put into actual practice. MYCIN system- and the project has served to substantially influence much of the sub sequent work in the construction and the implementation of expert systems.

The particular role proposed for MYCIN was that of providing assistance to physicians in the diagnosis and treatment of meningitis. Is and bacteria infections. MYCIN is thus somewhat akin to INTERNIST ICADUCEUS in its purpose, except. That it focuses on a far smaller number of diseases and thus requires a considerably smaller knowledge base.

The knowledge base of MYCIN contains the heuristic rules. EMYCIN (for empty MYCIN) is the name given to MYCIN when this specific knowledge base is removed. The result of incorporating acknowledge base associated with pulmonary disorders into EMYCIN resulted in a new expert system known as PUFF.

1.9.5. PUFF An Expert in Pulmonary Disorders

PUFF was developed using the EMYCIN shell. The purpose of PUFF is to interpret measurements related to respiratory tests and identify pulmonary disorders.

PUFF interfaces directly with the pulmonary test instruments used in such measurements. At the conclusion of the test, PUFF presents the physician with its interpretation of the measurements, a diagnosis of the illness, and a proposed treatment scheme. The first version of PUFF had 64 production rules. A more recent version had about 400 rules.

1.9.6. XCON (R1) An Expert in Computer Configuration

XCON (originally tilted R1) was developed of the configuration of VAX computers. AVAX computer may be configured in an enormous number of ways, and it attempts to configure each according to the specific requirements of each customer. XCON consisted of more than 8000 production rules running under the OPSS environment (a LISP based system that typically operates in a forward changing mode).

1.9.7. DELTE/CATS -An Expert in the Maintenance of Diesel- Electric Locomotives

DELTE/CATS-1 consists of knowledge base (i.e., set of heuristic rules) that was acquired through interviews. The system was originally developed in LISP and then converted to FORTH for increased transportability and speed of execution. Both forward and backward chaining is utilized.

A particularly interesting feature of DEL TE/CA TS-1 is its interface with visual support systems. More recently, remorse have circulated that DEL TE/CA TS-1 is having a problem similar to those cited for XCON, and the system may, in fact, have been shelved.

1.9.8. GATES-An Airline Gate Assignment and Tracking Expert System

GATES is in used evidently in prototype form. The system is being used to assist ground controllers in the assignment of gates to arriving flights. The knowledge base was acquired from an experienced ground controller who solved such problems on a daily basis.

The gate assignment problem can become quite complex, and requires rapid solution during intervals of flight delays, bad weather, mechanical failures, and so forth. Gates was developed, using the PROLOG, implemented on a personal computer.

1.9.9. OMR -Medical Diagnostic Expert System

Using the knowledge base first developed for INTERNIST, QMR assists physicians in the diagnosis of an illness based upon the patient's symptoms, examination findings, and Laboratory tests. QMR incorporates over 400 possible manifestations of diseases and is said to perform at level comparable to practicing physicians.

1.9.10. FXAA-Foreign Exchange Auditing Assistant

This involves thousands of transactions a day with paperwork resulting from such transactions weighing it at about 10 pounds per month. FXAA has been developed to provide the necessary auditing assistance. FXAA is a rule- based expert system that has evidently made a major or impact within Chemical Bank.

1.9.11. Jonathan's Wave -An Expert Commodities Trading

A number of firms and individuals have developed expert systems for stock and commodity trading. While it is still too early to asses the success of failure of these programs. Jonathan's Wave runs on two 286 based personal computers. The knowledge base is written in C while the inference engine is written in PROLOG. The system acts somewhat as through it was using multiple experts to reach its conclusion.

1.9.12. Insurance Expert Tax -An Expert in Tax Planning

Coopers and Libran have created insurance Expert Tax to assist in the identification of tax planning and accrual issues. Insurance Expert Tax took more than a year to develop and consists of more than 3000 rules. Created in LISP and running on the IBM PC.

1.9.13. HESS-An Expert Scheduler for the Petrochemical Industry

HESS was developed in support of product scheduling at a major petrochemical firm's refinery .The knowledge base in HESS was developed via the acquisition of heuristic rules from two refinery product schedulers. HESS was developed using the EXSYS expert system shell, through a 12-month effort. HESS, which stands for hybrid expert system.

1.9.14. An Expert Poultry Farming

They are developing an expert system for the poultry farmer. The system utilizes the Inexpert Object expert system shell. The system analyses data from the poultry farm's environmental control system. Using information on feed and water consumption, temperature, humidity, and ammonia levels, the system may be used to alert fanner to any diseases the chicken salve, or may get.

1.9.15. DUSTPRO-An Expert in Mine Safety

Using the Level 5 expert system shell as a development vehicle, it has developed an expert system named Dustpro. Dustpro replaces the limited number of human experts that assess the air quality of mining operations. Based on the amount of coal and silica dust in the air, mining operations must be adjusted to ensure that safety requirements are satisfied.

1.9.16. TOP SECRET-An Expert in Security Classifications

Within the Department of Energy (DOE), there are more than 100 classification guides to nuclear weapon security data. One of the more onerous tasks within the DOE is to attempt to correctly classify a given document though the use of these guides. Document classification determines who is permitted to view a document, and who is "not-a potentially critical factor in national security.

1.9.17.CODECHECK-An Expert in Computer Assessment

This expert system is for evaluation of codes. Termed Code check, the package is a rule-based expert system that checks C source code for such things as complexity, formatting, and adherence to standards.

The most common cause of hard to maintain software is the programmers, tendency to write overly complex code. Code check identifies those portions of the code may be simplified. In addition, Code check evaluates the portability of the source code by the comparing it with the numerous standards now existing for C programs.

1.9.18. Expert Systems for Faster, Fast Food Operations

Expert Systems have ever permeated the fast food market. A recent article describes the introduction of expert systems into such companies as McDonalds ...Here, such systems serve to reduce inventory, speed up service, and even act as training assistants. Packages provide valuable, timely assistance to managers who are neither familiar, not entirely comfortable with the pace of activities in such operations. In sector in which there exists such fierce competition, any improvement in cost reduction and enhanced operations can simply not afford to be overlooked.

1.10. Evaluation of Problem Types

Although just a few examples of expert systems have presented, we might note that they are preventative of the bulk of applications thus far developed. That is, the majority of applications involve classification (diagnosis). For example, in the medical expert systems, we are given certain data (symptoms) with regard to a patient and attempt to diagnose the associated cause, disease. In maintenance applications, precisely the same type of problem is faced. Here, the symptoms are the data on machinery performance while the diagnosis involves the identification of a detective or tailed component. Further, once a classification has been made, the specific class is matched to an associated treatment.

The remaining set of applications involves what is defined in this text as construction problems. XCON and HESS are representative of this type of application. Note that XCON attempts to construct AVAX computer, while HESS attempts to construct a schedule.

1.10.1 Classification & Construction Problems: Definitions

Classification as an attempt to draw boundaries about existing elements. For example, a certain set of existing symptoms point to a particular disease. Construction, on the hand, seeks to determine the arrangement of elements. That is, classification problems usually require backward search (Backward chaining) while construction problems typically require forward search (forward chaining).

Another, more visual, means for discriminating between these two fundamental types of problems is available by means of nothing just how each type of mental is mapped. To illustrate, consider figure below. On the Left of this figure, we have 5 objects. Associated with each value of these objects are certain attributes and values. On the right side of the figure, we have mapped these 5 objects into two groups.

To further clarify this concept, consider a problem in which the 5 objects on figure below are five different automotive engine parts. Each object is a set of data pertaining to various quality tests. Further, we simply wish to distinguish between parts that are acceptable and those that are not. Thus, a priori, we have two classes. Using the data set, a quality control engineer may then assign each object to one of the two Classes. And is a typical classification problem.

Next, let us assume that the problem involves the Loading of 5 items onto a fleet of trucks. Initially, we are not sure how many trucks are necessary. Associated with each item such attributes as weight, volume, cost, and priority. Using the values of these attributes, the cargo loader (or expert system) will then determine the loading scheme.

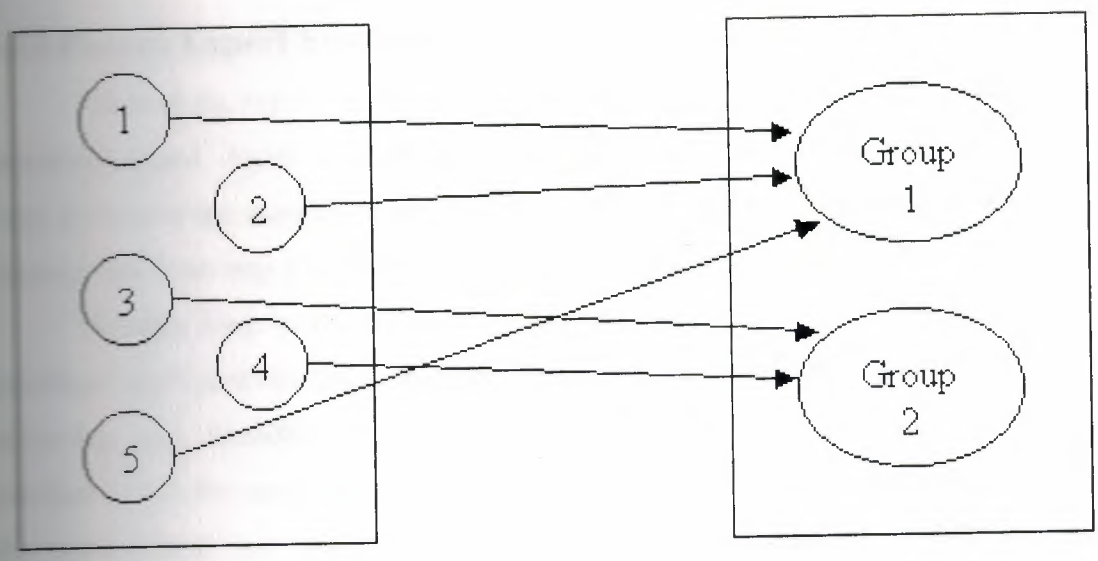


Figure 1.2. Mapping the Objects to grouping.

(Thus, above figure depicts a scheme in which items 1, 2, and 5 are loaded run one truck while items 3 and 4 are loaded on another truck. Again, note that the determination of the number of trucks used is an integral part of this problem, which is representative of a typical problem of construction.

A more recent, and more precisely defined, attempt toward problem Classification has been defined accomplished. He lists four types of applications for A! (Or expert system):

- Class l: Characterized by a need to select a solution from a fairly well defined set of possible alternatives -such as the medical diagnosis problem. This class coincides to what we have termed as classification problems.
- Class II: Characterized by a need to create a plan or configuration and scheduling. This class coincides to what we have tanned as construction problems.
- Class III: Characterized by need for the true creativity. Such problems include those of design~ including those where the very nature of the problem itself might have to be redefined.
- Class IV: Characterized as applications that humans can handle and computers can't. Included among this class are such problems as face recognition~ reasoning by analogy. And Learning how to talk.

2.ARCHITECTURE OF EXPERT SYSTEM

2.1 The Structure of System

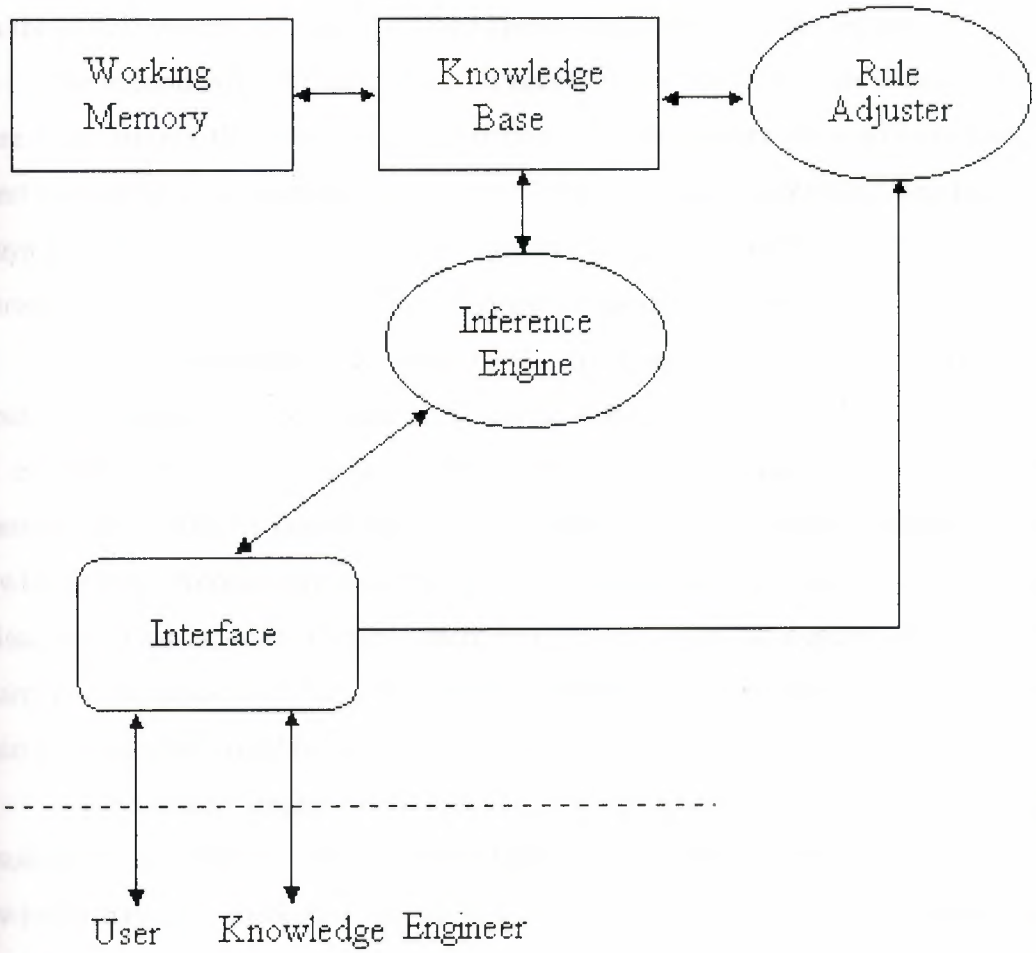


Figure 2.1. A generic expert system.

Figure 2.1 depicts one possible representation of an expert system. The components above the dashed Line are those with in the computer. Below this Line access capabilities of two types of human users are noted. The first is that individual designated as the knowledge engineer. As discussed, the knowledge engineer is the person responsible for placing into the expert system's knowledge base; the portion of the expert system shown at the top of figure 1. He or she accomplishes this through the *interface* and *rule adjuster*.

The knowledge engineer is also interface between human expert and the expert system. That is, the knowledge engineer somehow must capture the expertise of the human expert and then express this expertise in a format that may be stored in the knowledge base-and will be used by the expert system. In the ideal expert system, there would be no need for a knowledge engineer. The domain expert would interact directly with the expert system and would replace knowledge engineer in the figure

The second type of individual with access to the expert system is designated, in figure 1, as simply the user. This designation refers to anyone who will be using the expert system as a decision making aid. And the successful knowledge engineer must always keep in the mind that the expert system is ultimately intended for the benefit of the user, not for that of the knowledge engineer or the domain expert.

The interface handles all input to the computer and controls and formats all output. The interface would handle such scores. A well-designed interface would be one that exhibits ease of use, even for the novice user. The interface also handles all communication with the knowledge engineer during the development of expert system's knowledge base. Another property that sometimes exhibited in expert systems is that of explanation. That is, some expert system has limited ability to explain the reasons for the any questions asked of the user, as well as the rationale for the conclusion reached. Again this function would be the responsibility of the interface.

The interface engine is employed during a consultation session. During consultation, it performs two primary tasks. First, it examines the status of the knowledge base and working memory so as to determine what facts are known at any given time, and what facts are known at any time, and to add any new facts that become available. Second, it provides for the control of the session by determining the order in which inferences are made. An alternative designation for the inference engine and perhaps a more element appropriate one, is that of *knowledge processor.* As the knowledge-processing element of an expert system, the inference engine serves to merge facts with rules to develop, or infer, new facts.

The knowledge base is, as we have emphasized repeatedly, the very heart of any expert system. A knowledge base will typically contain two types of knowledge, that is, facts and rules. The facts within a knowledge base represent various aspects of a specific domain that are known prior to the exercise the expert system.

The working memory of an expert system changes according to the specific problem at hand. The contents of the working memory consist of facts. However unlike

the facts within the knowledge base, these facts are those that have been date 1 lined for the specific problem under consideration during the consultation session. More specifically, the results of the inference process are new facts and these facts are stored in the working memory.

The final module discussed in the rule adjuster. In most expert systems, this serves merely as a rule editor. That is, it enters the rules specified by the knowledge engineer into the knowledge base during the development phase of the expert system. It may also allow for various checks on these rules. In more ambitious expert systems, the rule adjuster may be used in an attempt to incorporate Learning into the process. In such instances, teach expert system by providing it with a set of examples and then critique its performance. If its performance is unsatisfactory, the-rule adjuster automatically revises the knowledge base. if satisfactory , the rule adjuster may simply reinforce the existing knowledge base.

An expert systems "shell" includes the entire components Listed figure 1 minus the knowledge base. Using a shell, it is up to the knowledge engineer to develop the knowledge base and to then insert knowledge base into the architecture to form a complete expert system, as intended for a specific domain. The use of a shell thus frees the knowledge engineer from the need repeatedly develop all supporting elements of a expert system, and thus the focus his or her attention on the development of the knowledge base.

The architecture of generic expert system, as depicted in figure 1, should serve to indicate at Least some of differences between this approach and that of algorithmic procedures, and heuristic programming. In particular note that the knowledge base is separated from the inference engine in other words, and unlike algorithm, and heuristic programming, an expert system separates heuristic rule, from the solution procedure. The knowledge base contains a description, or model, of "what we know". The inference engine contains, a description of "what we do" to actually develop the solution. While the knowledge base changes from domain to domain, the inference engine remains the same.

2.2. Inference Engine

The inference engine (IE) serves as the inference and control mechanism for the ES and, as such is an essential part of ES as well as major factor in the determination of the effectiveness and efficiency of such systems. Inference is the processes of drawing a

conclusion by means of set of rules, for a specific set of facts, for a given situation. Inference is thus the knowledge processing element of ES.

The most common inference strategy employed in ES is known as modus pones. Simply stated, modus pones means that if the premise of a rule is true, then its conclusion is also true. Thus if A infers B and A is true, then B is true. This may be represented as: A→B. Notice, however, that if B is true, we can not say that A is true. For example:

If an animal gives live birth

Then it is a mammal

However, we cannot say that if an animal is a mammal, it gives live birth.

Like the KB, IE contains rules and facts. However the rules and facts of the KB pertain to specific domain of expertise while the rules and facts of the inference engine pertain to the more general control and search strategy employed by the ES in the development of selection. These two sets of facts and rules are purposely kept separate in the typical ES. It is one of the key features of ES that serve to differentiate it from heuristic programming.

ES shells contain all of the necessary components with the exception of the KB. IE may work with different KB.

2.3. Search Strategies

The purpose of an ES is to develop and recommend a proposed solution to a given problem. To accomplish this task the ES must conduct a search for the solution, and it is responsibility of the IE in particular to perform this search in an efficient and effective manner. In the search process we are faced with a number of alternatives (i.e. potential solutions) and, typically a variety of constructions. For example, when faced with problem of determining just what automobile to purchase, we faced with certain constraints.

-Budgetary limitations.

-Automobile availability (i.e. not all automobile, not for sale within reasonable distance)

-Style. If we have family, we must have a car with 4 doors and a large storage area.

-Time. Our decision must be in certain interval of time.

Such constraints serve to filter out the number of potential automobiles from which we will make our selection. Other constraint factors are age of car, mileage on car, preference manufacturer, dealers, color and so on. Then we will focus on but a few cars from which we make our final selection. The search strategy implied in the selection of a car may be described in more technical terms as a forward chaining search with pruning. We begin the processes with certain data concerning the type of automobile desired, its style, cost, age, mileage, and so on. These data, along with our constraints, serve to filter out the majority of the potential alternatives and thus we arrive at only a few automobiles from which we make our final selection. The pruning processes reduces the size of the associated inference network. The inference network of figure serves to depict a simplified version of our process.

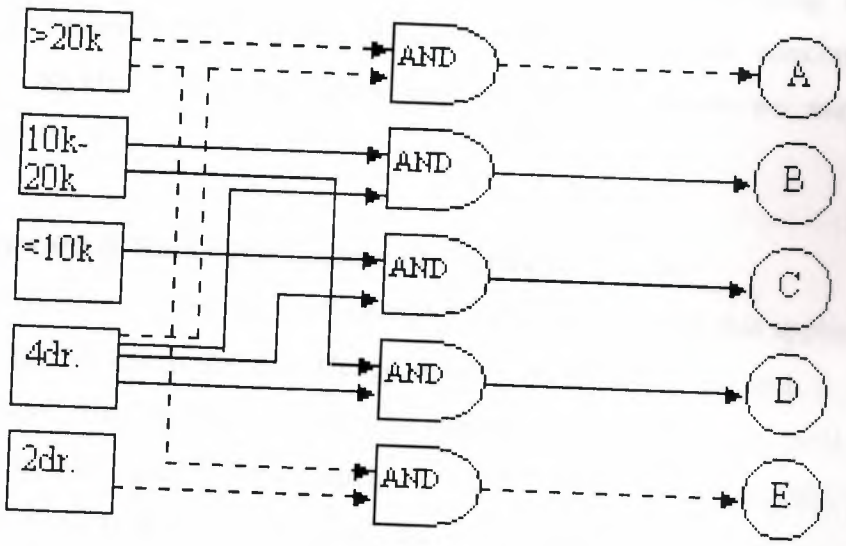


Figure 2.3. Inference network of automobile selection

The assertions in the boxes to the left represent

- Cost of automobile (i.e., greater than $20.000; between $10.000 and $20.000; and less than $10.000)
- Number of doors (i.e., 2 door or 4 door)

The conclusion, to the right, represent automobile choices (A, B, C, D, and E). Now, if we restrict ourselves to cars that are no more than $20.000 in cost and that have four doors, we can reduce the inference network in size. Specifically, we may eliminate the dashed branches as well as all assertions, conclusions, and logical connectives associated solely with these branches.

We could approach our automobile selection problem from an entirely different direction by first specifying a particular car for purchase, and then determining whether or not it meets our needs. When using this approach, we are said to employing backward chaining. For example, we might first consider the purchase. Having established this as tentative decision, we then determine whether or not it is feasible (i.e. does it satisfy the KB).

The two fundamental search strategies employed by an ES: forward and backward chaining. Forward chaining proceeds from premises (or data) to conclusions, and is said to be data driven. Backward chaining proceeds from tentative conclusion backward to the premises to determine if the data supports that conclusion. Backward chaining is often called a goal-driven approach and proceeds from right to left. If one has a few premises and many conclusions then forward chaining is best chaining strategy, otherwise with many premises and relatively few conclusion we should employ backward chaining. There are instances in which we may employ both approaches.

2.4. Forward & Backward Chaining In Inference Engine

The inference engine is the generic control mechanism that applies the axiomatic knowledge present in the knowledge base to the task-specific data to arrive at some conclusion. This is the second key component of all expert systems. Having a knowledge base alone is not of much use if there are no facilities for navigating through and manipulating the knowledge to deduce something from it.

As a knowledge base is usually very large, it is necessary to have inference mechanisms that search through the database and reduce results in an organized manner. A few techniques for drawing inferences from a knowledge base are described here.

2.4.1 Forward Chaining

Consider the following set of rules

Rule 1: IF **A** and **C** THEN **F**

Rule 2: IF **A** and **E** THEN **G**

Rule 3: IF **B** THEN **E**

Rule 4: IF **G** THEN **D**

Suppose it needs to be proved that **D** is true, given **A** and **B** are true. Start with Rule 1 and go on down till a rule that "fires" is found. In this case, Rule 3 is the only one that fires in the first iteration. At the end of the first iteration, it can be concluded that **A,**

B and **E** are true. This information is used in the second iteration. This time Rule 2 fires adding the information that **G** is true. This extra information causes Rule 4 to fire, proving that **D** is true.

This is the method of forward chaining, where one proceeds from a given situation toward a desired goal, adding new assertions along the way .In expert systems, this strategy is especially appropriate in situations where data are expensive to collect, but few in quantity.

2.4.2. Backward Chaining

In this method, one starts with the desired goal, and then attempts to find evidence for proving the goal. Returning to the previous example, the strategy to prove that **D** is true would be as follows.

First, find a rule that proves **D**. Rule 4 does so. This provides a sub-goal -to prove that **G** is true. Now rule 2 comes into play, and as it is already known that **A** is true, the new sub-goal is to show that **E** is true. Here, Rule 3 provides the next sub-goal of proving that **B** is true. But the fact that **B** is true is one of the given assertions. Therefore, **E** is true, which implies that **G** is true, which in turn implies that **D** is true.

3.KNOWLEDGE REPRESENTATION

3.1. Models Of Knowledge Representation

The knowledge that is contained with the an expert system consists of:

- *A priori know/edge*: the facts and rules that are known about a specific domain prior to any consultation session with the expert system.
- *Inferred know/edge*: the facts and rules concerning a specific case that is derived during, and at the conclusion of, a consultation with the expert system.

The major goal is to represent the facts and rule within the knowledge base of expert system. For this reason, it is needed to:

- Provide a format compatible with the computer.
- Maintain as close as possible a correspondence between this formats and actual facts and rules.
- Establish a representation that can be easily addressed, retrieved, modified, and updated.

Elaborating further on the last two points, it would be highly desirable to use a format that is *transparent*, that is, a representation scheme that may be easily read and understood by humans.

3.2 Object -Attribute -Value Characteristics

Alterative modes of knowledge representation is:

- OAV {object-attribute-value) triplets
- Semantic networks
- Frames
- Neural networks

3.2.1. OAV Triplets

Object-attribute-value triplets provide a particularly convenient way in which to represent certain facts within a knowledge base and may be extended (as we shall see) to provide the basis for the representation of heuristic rules. Each OAV triplet is concerned with some specific entity, or object. For example, our object of interest might be an airplane. Associated with every object is a set of attributes that serve to characterize that object. Using the airplane as an example (i.e., as the object), some of its attributes include the following:

- Number of engines

- Type of engine {e.g., jet or prop)
- Type of wing design (e.g., conventional or swept black)

Figure 3.1. OAV network

For each attribute, there is an associated value, or set of values. For instance, in the case of the C130 military cargo aircraft (known as the Hercules), the number of engines is four, thf ! jty- of engine is prop, and the wing design is conventional. Notice in particular that values in OAV triplets may be numeric or symbolic. We may List these facts as shown below.

Number of engines = 4

Engine type = prop

Wing design = conventional

Observe that, in this List, the object itself (i.e., the C130 aircraft) is never explicitly stated. Actually, the above statements represent AY (attribute-value) pairs. ! However, associated with any AY pair is some object. Thus, any AY pair implies an OAY triplet.

Yet another way *to* represent an OAY triplet would be through the use of a network representation as indicated in Fig. 3.1. The basic building blocks of a network are its nodes (i.e., the circles) and branches, or edges (i.e., the lines connecting two nodes). In Fig. 3 .1, the object is Pete Jones, the attribute is his income, and the specific value of his income is $50-000

3.2.2.Semantic Networks

A semantic network may be thought of as a network that is composed of multiple OAV triplets in network form as illustrated in Fig. 3.1. However, rather than 1. Pertaining to just one attribute for a single object, semantic networks may be used to

represent several objects, and several attributes per object. Returning to our aircraft illustration of the previous section, we might develop a partial semantic network as illustrated in Fig. 3.2, here, we note that the CSA is a special type of aircraft (i.e., a large military cargo plane). Further, since the CSA is an aircraft, it *inherits* the properties associated with aircraft in general (e.g., it flies, has wings, carries people). Such an inheritance property can prove to be of considerable value in the reduction of memory storage requirements. That is, since a CSA is an airplane; there is no need to store, at the CSA node, the fact that it can fly, has wings, and can carry people. Thus, the semantic network scheme provides for a convenient approach for the representation of *associations* between entities.

We might also note that the OAV triplet is actually just a restricted subset of semantic networks wherein the only relationships that may be used are those of "is-a " and "has-a" OAV nodes, in turn, may be any of three types; objects, attributes, or values.

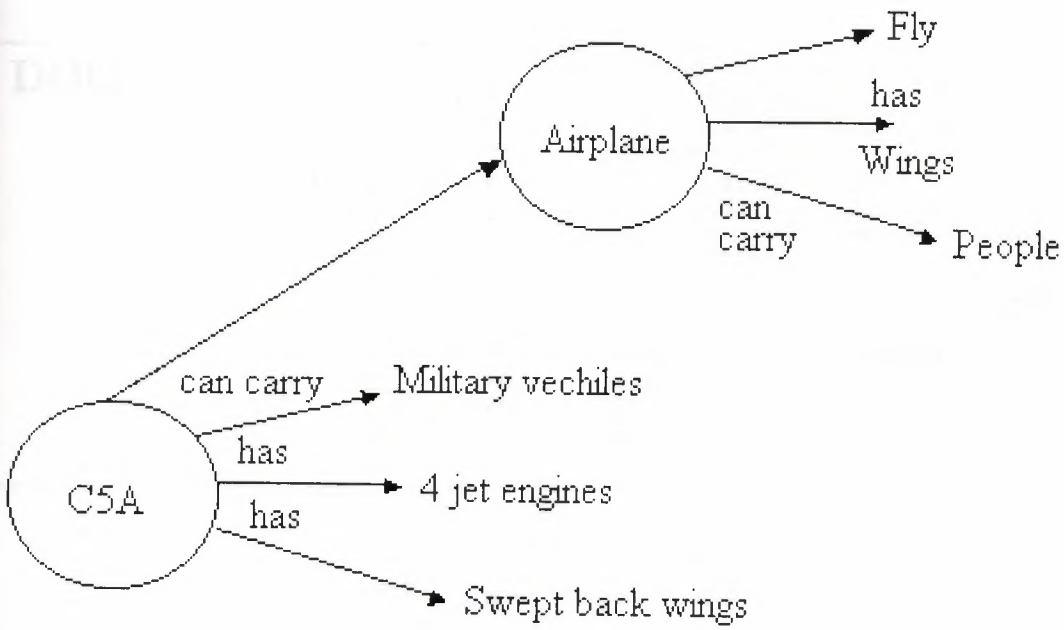


Figure 3.2. Semantic Network

3.2.3. Frames

While semantic networks provide a relatively versatile means for knowledge representation, the use of frames represents an alternative approach that serves to capture most of the features of the semantic network while providing certain additional aspects. In fact, we may think of a semantic network as being a subset of the concept of frames.

The employment of frames represents a particularly robust way in which to present knowledge. A frame contains an object plus *slots* tar any and all information related to the object. The contents of such slots are typically the attributes, and attribute values, of the particular object. However, in addition to storing values for each attribute, slots may contain default values, pointers to other frames, and sets of rules or procedures that may be implemented.

Figure 3.3 illustrates a frame-based representation for the object dog. Note that the slots within this frame include values (e.g., Beagle), defaults (e.g., four Legs), and procedures (e.g., for a medical examination). The procedures, in turn could well point to other frames. The versatility of the frame-based mode of knowledge representation should be obvious.

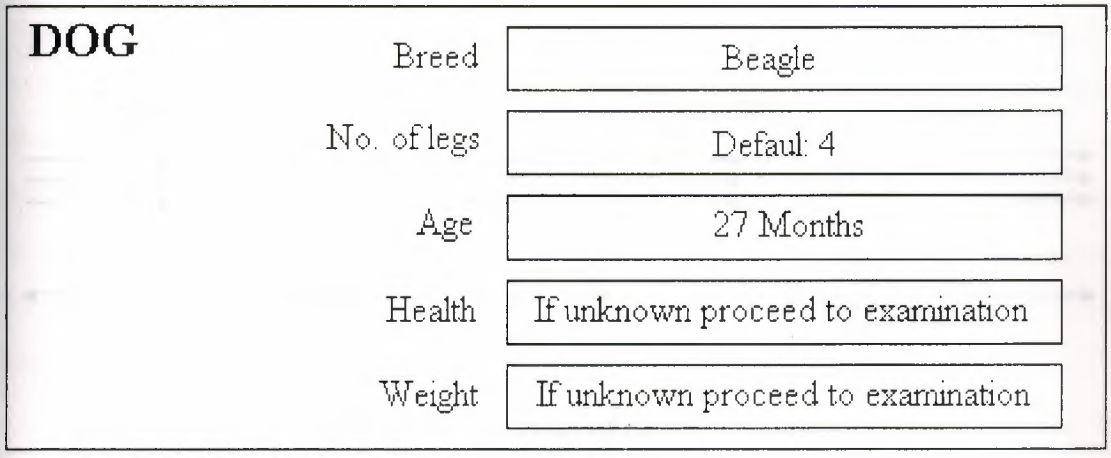


Figure 3.3. A frame based representation

The primary drawback to the use of frames is, ironically, caused by the very robustness of such a mode of representation. Frames. Have so many capabilities as to make their use a rather complex matter. Jackson [1986] states that "many people are unhappy with frame- and object-based systems because they seem to departure Logic

and because their flexibility in matters of context and control can make their behaviou both hard to predict and difficult to understand. " As a result, to obtain any reasonable proficiency in the use of frame-based tools in expert systems a Lengthy training period is required. Despite such drawbacks, frames can prove quite useful, if not essential, in the design of Large-scale, complex expert systems-particularly those involving a Large

Amount of a priori facts (i.e., data), and multiple objects. While frames are not focused on in this text, it is strongly encouraged that the serious student investigates this *topic-after* he or she has attained a reasonable Level of competence in the use of rule bases,

3.2.4. Neural Networks

Obviously, somehow, some way, the human brain stores knowledge. What is not t so obvious is the precise manner in which this is accomplished. Neural networks represent mankind's attempt to replicate, in hardware, theories pertaining to the it brain. Specifically, it is thought that knowledge is stored in neurons (or, actually, in -the connections between neurons). Figure 3.4 depicts a simplified representation of only two neurons within the neural network of the human brain.

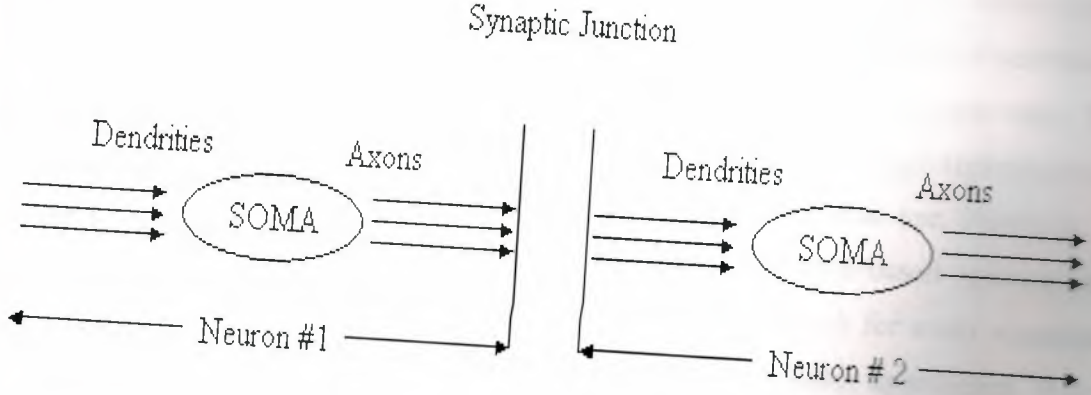


Figure 3.4. A portion of neural network

In the human brain there are more than 10 billion neurons, and each neuron is connected to one or more other neurons, resulting in a massively interconnected network. At each neuron, impulse $ are received by the dendrites and transmitted; i by the axons. If the output of the axon is at a *high-enough* Level, the signal will *jump* the synaptic junction and trigger the connected neuron (or neurons) .It is believed that knowledge might then be represented by the weightings on each neuron to neuron

interconnection, which in turn influence the Level of strength of the interconnecting impulses.

The attempts to duplicate the neural network structure of the brain have been, at best, extremely modest. Typically, electronic amplifiers are used to represent the neurons and resistors to correspond to the interconnecting weights, and existing systems have but a few Layers of relatively few neurons. Despite this, neural networks can be used to accomplish some intriguing tasks, including some success in speech recognition. In particular, they provide a robust approach to the general problem of pattern recognition. Probably the biggest single disadvantage of the neural network approach to knowledge representation is the fact that any knowledge that exists is almost totally opaque.

Since neural networks are often excellent choices for problems of classification, they may be combined with an expert system to perform certain tasks. That is, the neural network may be used to classify and, based upon this class, the expert system may then be used to determine the specific course or courses of action to take.

3.3 Representation via Rule-8ased Systems

Undoubtedly, the most popular mode of knowledge representation within expert systems, at lest at this time, is the mode obtained through the use of rules, or rule-based systems. Alternatively, such rules are referred to as IF-THEN, or production rules. We have selected rule-based expert systems as our approach to knowledge representation for a number of reasons, including this popularity and widespread use. However, it should be stressed that this decision does not imply that rule-based systems are necessarily the best approach or , in particular , the best approach for every situation. There are those who present quite persuasive arguments for other approaches. Rule-based knowledge representation has been made for the following reasons:

- The majority of existing expert systems development packages employ rule based.
- Rule-based expert systems development packages are normality much Less expensive (in terms of both the initial cost of the package as well as the overall cost of using the package) than those employing alternative modes of representation. Specifically. They cost less to purchase, normally do not require any expensive hardware (most runs on inexpensive, general purpose personal computers), and require minimal expenditures toward training.

- The widespread availability of rule-based expert systems shells permits the knowledge engineer to focus his or her attentions o the most critical phase of the development of an expert system, that is, on the knowledge base.
- Rules represent particularly natural mode of knowledge representation. '" Consequently, the time required to learn how to develop rule bases is minimized.
- The learning curve for rule-based expert systems is much steeper than for any alternative mode of representation.
- Rules are transparent, and are certainly far more transparent than the modes of knowledge representation employed by rule-based systems two major competitors: frames and neural networks. Further, such transparency often leads to an increased willingness, on the part of management, to accept the solutions obtained. And the importance of this last factor should be underestimated.
- Rule bases can be relatively easily modified. in particular, additions, deletions, and revisions to rule bases are relatively straightforward processes. And this is particularly so in the case of well designed rule bases.
- Rule based expert systems can be employed to mimic most features of frame - based representation scheme.
- Validation of the content of rule-based systems is relatively simple process. Similar validation of frames or neural networks, on the other hand, is normally difficult to impossible.

3.3.1. Production Rules: An Overview

Rule-based modes of knowledge representation employ what are termed production rules or, for short, simply rules. Such rules are typically of the IF-THEN variety. However, in some instances this is extended to include IF-THEN.ELSE rules. For example, we might have the If. THEN-ELSE rule as Shown below:

Rule 1: If the student's score ORE score is 1350 or more then admit the student to the graduate program Else, do not admit the student

Which is equivalent to two IF -THEN rules or,

Rule 1 a: If the student's score ORE score is 1350 or more then admit the student to the graduate program

Rule 1 b: If the student's score ORE score is less than 1350 then do not admit the student For the clarity of presentation, we shall focus primarily on just IF-THEN rules.

In fact, it is generally advisable to avoid the use of ELSE statements in rule-based expert systems. This is true for three reasons. First, a number of commercial expert systems development packages simply do not permit the use of the IF-THEN - ELSE rules. Second, validation of such rules is considerably more difficult than for their IF -THEN equivalents. Third, when encountered in the inference process, such rules will tend to always reach conclusion. This can result in some unanticipated results. Thus, whenever one comes upon such a rule, we strongly advise the formation of the corresponding two equivalent rules.

An alterative designation for IF-THEN rules is that of *condition-action* or *premise-conclusion* statements, we shall refer to the IF statement as the premise and to the THEN statement as the conclusion,

We should also realize that there might be several premise and conclusion statements within a single rule. Each of these is tended clauses (i.e. premise clauses and Conclusion clauses). Another rule with multiple clauses is in the IF and THEN portions. Further, not that while premise clauses may be connected by a AND as well as OR operators, the conclusion clauses may only be connected by AND statements. That is, all of the conclusion clauses in a production rule must be true.

Clauses connected by AND operators are denoted as conjunctive clauses. Those connected by OR operators are tanned disjunctive clauses.

3.3.2 Attribute Value Properties

As noted, each premise and conclusion clause contains attributes and values. Further, there must be an associated object, either implied or explicit. Consider, the rule shown below:

Rule 1: if grade point average (GPA) equals or exceeds 3.5 then accept into honour society

When clauses contain only attributes and values, as in the case of the rule under discussion, they are sometimes called attribute -value or A V pairs. In the conclusion clause, the attribute-value pair is accepted into honour society. Actually, this is a poor choice of wording for this conclusion. In general, rules should be written so that identification of the attribute and value is straightforward -while the rules remain intelligible.

The A V pair is the fundamental building block of a premise or conclusion. And thus the fundamental building block of a production rule. Associated with each A V pair is a set of properties. The most typical of these are below:

- **Name:** The name of the attribute is simply the wording selected to identify the attribute of the object associated with the clause under the consideration. For example, some of the attributes typical of the automobile are colour, number of doors.
- **Type:** Attribute values may be either Numeric or symbolic. For example, the temperature of a patent may be given in degrees Fahrenheit- a numeric value. Alliteratively, we might specify the temperature values to be symbolic, such as high or normal. Yet another symbolic values would be yes or no, for example, such as with respect to the presence or absence of some feature.
- **Prompt:** Associated with certain attributes are user prompts, or queries. When necessary , the user replies to this prompt with a value for the attribute under consideration. The only attributes that should normally be provided with Prompts are:

i. Attributes that appear in a premise statement and never appear in any conclusion statement of the rule set

ii. Attributes for which the user can conceivably provide a response.

- **Legal Values:** Associated with every attribute is a set of legal or acceptable, values. For example, the Legal for the person's weight would simply be the set of nonnegative real numbers. If the replies with a no legal value, this is detected and the user may be asked to reply again. In the case of expert systems that provide menu-driven prompts the set of Legal values is simply presented to the user and he or she can only select from that List.
- **Specified Values:** indicate the actual set of values that are either to be tested against (in a premise clause) or that will be, or have been, assigned (in a conclusion clause). More specifically, we are consumed with whether or multiple specifications are permitted. Multiple values may also be allowed (where, again, this is dependent upon the particular software package employed) , for attributes that .appear in conclusion clauses. In other words, it may be permitted to assign (i.e. conclude) multiple values to the attribute in a conclusion clause.

- **Confidence Factors:** If the expert systems development package permits, we may deal with uncertainty in either conclusions (i.e. the conclusions attribute values assigned) or premises (i.e. the premise attribute values used). Since we shall not deal with uncertainty and confidence factors in this chapter, we shall merely note that this too is an A V pair property.

3.3.3 Clause properties

As we discussed, there are two types of clauses: premise and conclusions. Other properties associated with clauses are in the below list:

- Single versus multiple (or compound) clauses
- Conjunctive versus disjunctive (multiple) clauses
- Free (premise) clauses
- Specified (premise) clauses (i.e. Specified true or Specified false)

Let us examine each of these properties in turn. First, a premise or conclusion may consist of a single clause or set of clauses.

Multiple clauses. In tern may be either conjunctive causes (each cause connected by tilde AND operator) or disjunctive (each clause connected by the OR operator) 1 however, recall that disjunctive clauses are not palliated in the conclusion of a rule. Also, note the premise of a rule may be quite complex.

Another property oaf clause is that associated with premise clauses only. This is the property of being either free of specified, and if specified, of being true or false. If the value of premise clause attribute is not yet known, that clause is designated as a free clause. Note most carefully that we have drawn a distinction between not yet known and unknown. If a clause is not free, then such a clause is either true or false. Consider the following simple premise clause shown below:

If A=X

Note must be attribute for the object, which the clause is consumed. X is then one possible (legal) value for this attribute; we must test this clause to see if **A** does indeed equal X it we do not know the value for **A**, and have yet to seek this value, the clause is free. However, if we do know the value for **A**, and this value is indeed X, then clause is true. Other wise, (i.e. if the value of A is known but is something other than X), the clause is false.

The properties of free, true, or false would seem be straightforward. And indeed they are; however a certain degree of confusion may occur when one employs unknown

an attribute value. Note carefully that we must differentiate between known and not known yet. Not known yet means that the value for a respective attribute has not yet been determined. Thus the associated clause is free.

Unknown, however, can be employed in one of two ways.

i. It may simply be a legal value for a given attribute. The premise clause is true, and the rule is triggered.

ii. Unknown may be employed as slightly more complex~ and a function of the specific mode of the inference used by the software package. In this case a vague of unknown is assigned to an attribute whenever its value cannot b determined from the inference procedure.

3.3.4. Rule Properties

As with A V pairs and clauses, there are certain important rule properties. Some of the more typical rule properties are below:

- **Name:** Each rule should have a distinct, as well as descriptive name. Actuality, we have not always followed this guideline. However, it is a good idea to do this when building any actual knowledge base. Specifically, rather than just Labelling a rule by a number or letter.
- **Premise:** Every rule consists of one or more premise clauses is termed the rule premise. A rule premise may consist of conjunctive or disjunctive clauses.
- **Intermediate Conclusions and Conclusions:** Every rule consists of one or more conclusion clauses. In the case of multiple conclusion clauses, the clauses must be conjunctive. There are two types of rule conclusions: intermediate conclusions and (final) conclusions. An intermediate conclusion is one that does not appear as a premise clause for any other rule.
- **Notes & References:** It is essential that a rule base be documented. While you, the developer, may know the reason and source of the rules, others will not. Further, with the passage of time, even the developer will find it difficult to recall the origin and specifies of each rule. Many development packages permit the inclusion of notes and references; add this is a feature that should most definitely be employed in any actual knowledge-base development.
- **(Rule) Confidence Factors:** When uncertainty, s employed, we may associate confidence factors with each rule. Here, we may simply note that the

confidence factor of a rule's conclusion is a function of the confidence factors of the rule and the rule premise.

- **Priority & Cost:** In some development packages, we are permitted to assign a priority and/or cost to each rule. Such properties are normally employed as a means to decide, during the inference procedure, this specific rule to be dealt with at a particular instance. Typically, the procedure will select the rule with the highest priority or the lowest cost.
- **Chaining Preferences:** The inference process involves a search procedure. In some cases, the search moves forward direction-from premises (or facts) to conclusions. In other, the search moves, backward-from a hypothesized conclusion to the premises necessary to infer that conclusion. However, in addition to such normal modes of search, or chaining, some development packages permit the employment of a mixture of search methods.
- **Rule Status:** During consultation, the status of each clause and rule's subject to change. Keeping track of such changes is an essential part of the inference process. We need to become acquainted with the terminology used. A summary of this terminology's provided below:

i. The premise of' a rule is true whenever a test has been made and it has been determined that the premise has been satisfied.

ii. The premise of a rule is false whenever a test has been made and it has been determined that the premise has not been satisfied.

iii. If the premise of a rule is true then that rule is said that be triggered.

iv. If the premise of a rule is false then that rule may be discarded or, in some cases, made inactive.

v. If a rule is fired then this implies that action implied by the conclusion clause(s) is taken. The values associated with each attribute of the conclusion clauses for this rule are said that to be assigned.

vi. A rule that has been fired is no longer active. It is either discarded or, in some cases, made inactive.

vii. If a rule is to be fired, that rule must be first have been triggered.

viii. If a rule has been neither fired nor discarded, that rule's designated as being Active.

3.3.5 Rule Conversion: Disjunctive Clauses

While such conversions are not necessary in the general. Methodology of expert systems, they often make easier for the beginner to follow the inference process of an

expert system when manual demonstrations are employed. Further some expert systems development packages do not permit the use of disjunctive premise clauses. However, such a conversion does result in an enlargement of the number of rules necessary to represent the knowledge base of an expert system. Despite this the beginner may be well advised to consider such a conversion -as well as determine the restrictions of the software that is to be used.

3.3.6 Multiple Conclusions

We must stress, however that it may be quite reasonable for an expert system to draw multiple conclusions and this is particularly so if we are dealing with the uncertainty.

There are also instances in which multiple conclusions may make sense even though uncertainty is not being employed. T o illustrate, consider the three (deterministic) rules listed below:

Rule A: if client's risk profile is risk adverse

Then client's investment strategy is blue chip stocks

Rule B: If client's investment portfolio is less than $50,000

And client's age is more than 60

Then client's investment strategy is high-grade bonds

Rule C: if client's risk profile is risk taker

And client's age is Less than 45

Then client's investment strategy is growth stocks

In essence, we have concluded that either strategy is advisable. Thus in this case, of deterministic rule bases, the validity of multiple conclusions is a function of the situation. Again, however, realize that not all development packages permit multiple conclusions.

4. KNOWLEDGE ACUISITION

Definition of knowledge acquisition is the transfer and transformation of potential problem solving the expertise form some knowledge source to a program. Knowledge acquisition is a generic term, as it is neutral w1th respect to how the transfer of knowledge is achieved. The knowledge elicitation, on the hand, often implies that transfer is accomplished by a series of interviews between a domain expert and a knowledge engineer that then writes a computer program representing the knowledge. The term could also be applied to the interaction between an expert and a program whose purpose is:

- To elicit knowledge from experts in some systematic way.
- To store knowledge so obtained in some intermediate representations.
- To compile the knowledge from the intermediate representation into a run able form, such as production rules.

The use of such programs is advantageous because it is Less Labour intensive, and because it accomplishes the transfer of knowledge from the expert to a prototype in a single step.

4.1 Stage Of Knowledge Acquisition

It is worth summarizing these stages are here:

- **Identification**: Identify the class of problems that the system will be expected to solve, including the data that the system w111 work w1t~ and the criteria that solutions must meet. Identify the resources available for the project, in terms of expertise, manpower, time constrains, computing facilities and money.
- **Conceptualisation:** Uncover the key concepts and the relationship between them. This should include a characterization of the different kinds of data, the flow of information and the underlying structure of the domain, in terms of causal spatial-temporal, or part-whole relationships, and so on.
- **Formalization:** Try to understand the nature of the underlying search space, and the character of the search that will have to be conducted. Important issues include the certainty and completeness of the information, and other constraints upon the Logical interpretation of the data, such as time dependency, and the reliability and consistency of the deterrent data sources.

- **Implementation:** In turning a formalization of knowledge into run able program, one is primarily consumed with the specification of the control and the details of information flow.
- **Testing:** The evolution of expert systems is far from being a exact science, but it is clear that the task can made easier if one is able to run the program on a large and representative sample of test cases. Common safes of error are rules, which are ether missing, incomplete, or wholly incorrect, while competition between related rules can be, cause unexpected bugs.

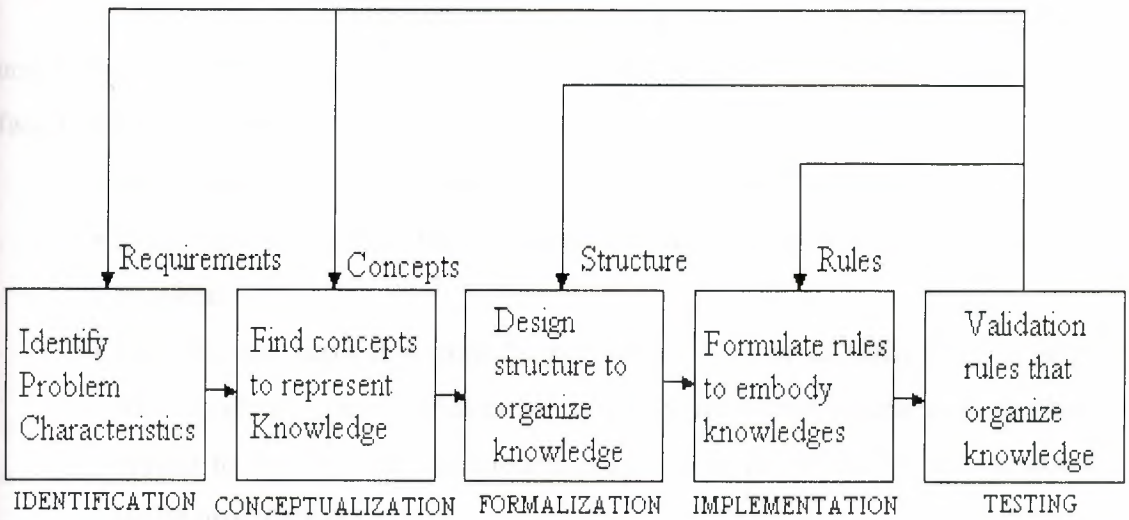


Figure 4.1. Stages of knowledge acquisition

4.2 Different Levels In the Analysis of Knowledge

The distinction drawn between identification, conceptualization and formalization can also be found who have developed a modeling approach to knowledge engineering within frame called KADS. The authors argue that a knowledge-based system is not a container filled with knowledge extracted from an expert but an.! Operational. Model that exhibits some desired behavior and impacts real. World phenomena. Knowledge acquisition involves not just eliciting domain knowledge but also interpreting the elicited data with respect to some conceptual framework and formalizing these conceptualizations in such a way that a program can actually use the knowledge.

- The static ontology, which consists of domain entities, together with their properties and relations.
- The dynamic ontology, which defines the states that occur in problem solving, and the manner in which one state may be terms formed into another.
- The epistemic ontology, which describes the knowledge that guides and constrains state transformations.

There is Less of correspondence with Lower Levels, such as the Logical and implementation analysis.

4.4. Expert System Shell

Early expert systems were built "from scratch", in the sense that the architects either used the primitive data and control structures of an existing programming Language to represent knowledge and control its application, or implemented a special purpose rule or frame Language in an existing programming Language, as a prelude to representing knowledge in that special purpose Language.

- Modules, such as rules or frames, for representing knowledge.
- An interpreter, which controlled when such modules became active.

The modules, taken together, constituted the knowledge base of the expert system, while the interpreter constituted the inference engine. In some cases, it was clear that these components were reusable, in the sense that they would serve as a basis for other applications of expert system technology. Since such programs were often abstractions of existing expert systems, they became known as expert system shells.

4.5. Knowledge Acquisition Methods

One involves knowledge acquisition for troubleshooting a telephone company switching system, and the other involves planning therapeutic regimes for cancer patients. The two projects dealt with the issues of-knowledge acquisition and knowledge representation in rather different ways, largely as a consequence of both the task at hand and the way that the task was approached.

4.5.1 Knowledge election by interview in compass

A telephone company,, switch" is not simple device, but extremely complex system whose circuitry may occupy a Large part of building. The goals of switch maintenance are minimize to the number of calls that have to be rerouted owing to bad connections and ensure that faults are repaired quickly to maintain the redundancy of

the system. Bad connections are caused by some failure in the electrical path through the switch that connects two telephone Lines.

COMPASS is an expert system, which examines error messages derived from the switch's self test routines, which Look for open circuits, short, Lag time, in the operation of components, and so forth. Looking at a series of such messages and bringing significant expertise to bear can only identify the cause of a switch problem. *COMPASS* can suggest the running of additional tests, of the replacement of a particular component, such as a relay or circuit card.

The knowledge acquisition cycle employed in COMPASS had the following form:

1.Elicit knowledge from the expert.

2. Document elicited knowledge.

3. Test the new knowledge as follows:

- Have the expert analyse a new set of data
- Analyse the same data in a hand simulation using the documented knowledge.
- Compare the results of the expert's opinion with the hand simulation.
- If the results differ, then find the rules or procedures that generated the discrepancy, and the return to (I) to elicit more knowledge from the expert to resolve the problem, else exit loop.

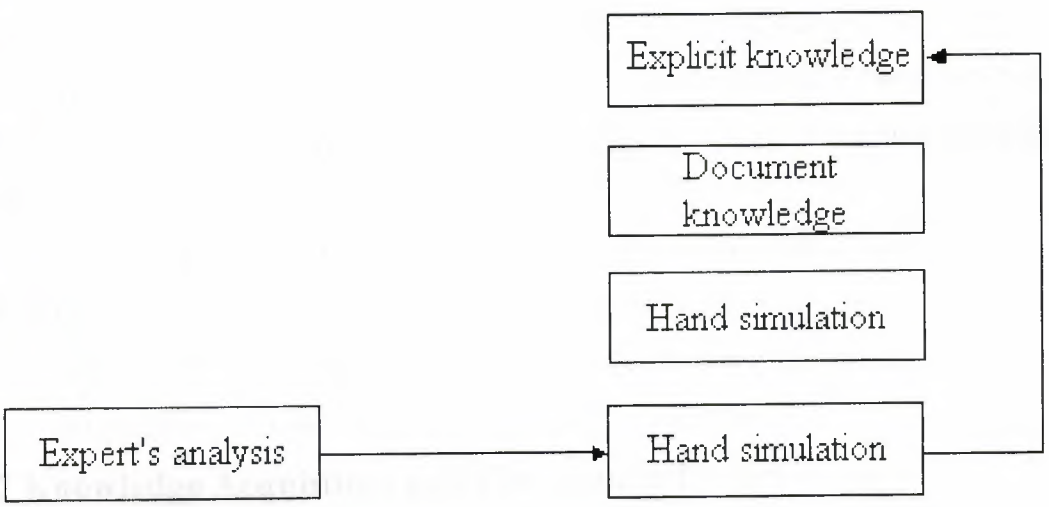


Figure 4.2. Knowledge acquisition cycle in COMPASS.

This elicit-document-test cycle is represented graphically in above figure.

4.6 Knowledge- Based Knowledge Acquisition

We shall see that many of the Learned from trying to extend expert system technology in various directions have application to knowledge acquisition problem. Specifically,

- Attempts to use expert systems as a basis for indigent tutoring systems have led to a dear understanding of the different kinds of knowledge that expense deploy in problem solving; and
- Attempts to build generic expert system tools like EMYCIN have posed interesting problems consuming how to help developers with the task of encoding knowledge from some arbitrary domain into frame or production rule format.

Such endeavours have required researchers to examine the role of domain knowledge and domain inference more closely, particularly with respect to the different styles of reasoning that are approximate to different domains.

Looking ahead slightly, what seems to be clear is that knowledge acquisition is greatly facilitated by being itself knowledge based. In other words, a knowledge elicitations programs needs some knowledge of a domain or a problem area in order to acquire new knowledge effectively, just as knowledge engineers need to have some knowledge of a domain before they can communicate effectively with an expert.

Perhaps this result is not. Surprising, given the Lessons of knowledge-based approaches to problem solving. Knowledge elicitation is a substantial problem in itself, and there is no reason to suppose that. There is a single general method that will be effective in all domains~ any more than there is reason to suppose that there are general problem solving methods that will always be effective.

Knowledge elicitation model by interview based on a domain model is not the jest. Word in automated approaches to acquisition. We shall two further approaches:

- Acquisition strategies organized around a particular problem solving method.
- Unsupervised machine Learning of roles by induction.

4.7 Knowledge Acquisition and The domain Expert

It wound seem that the most Obvious in which one may acquire knowledge base is to go directly to the human expert. However, there are at feast. Four reasons why this may not work, or at feast not provide totality satisfactory results. For some problems, there simply may not be an expert. One example that comes to mind is that of investing

in the stock market. While some inventor or investment advisory services do well for a relatively brief time. They typically have spells in which their performances mediocre to terrible.

- To allege experts may actuality be exhibiting poor mediocre performance. AU too often, the term expert is foolery applied to anyone who simply gets the job done.
- The expert may not wish to reveal their tricks of the trade. In some cases, such individuals simply refuse to cooperate. In others, potentially far more serious problems occur.
- The experts may not wish who are just unable articulate the approach that they use. Many experts, in fact, simply and honestly do not really understand how they actually make their decisions.

However, based upon the assumption that a human is performing the task that is to be performed in the future by the expert system, our first step is to identify that person. Once this person has been identified, the next step is to set up to an initial. Meeting with the alleged domain expert. This meting should be infernal because you must decidedly want the inaugural meeting with this person to take place in a relaxed atmosphere.

There are several purposes for the initial meeting. First, we wish to relax the individual. Second, we should attempt to explain to the individual just precisely what it is that we intend to accomplish. Typically, one emphasizes that our purpose is not to use and then discard the expert, but rather it is to provide him or her with a computerized assistant so that he or she can pursue more interning work.

In certain cases, the initial meeting should be followed, or even preceded by an on site visit. There is simply no substitute for actually being able to view the problem in its physical context.

There is yet another purpose to the initial meeting, as well as those that follow, which should be openly discussed. Specifically, we should use this meeting, as well as follow-on meetings, to attempt to evaluate the true extent of the expertise of our expert. If and when you encounter a domain expert in whom you have no confidence, there are number of alternatives that should be considered. First, and most obviously, you might try to find a replacement someone else in the organization that seems reasonably competent in the domain under consideration. This option of course, requires a certain amount of diplomacy. Second, you might consider Learning enough about the problem

at hand so that you can act as the domain expert. Third, might wish to examine historical records of decision made.

Returning to primary point our discussion, as Long as feel confident that the expert systems approach is indeed the most appropriate, and the domain expert is reasonably competent, we may continue with the knowledge acquisition process. Thus, following initial informal meeting with the domain expert, we should conduct a series of formal meetings designed to extract as much information.

The conduct of the follow-on meetings is generally best handled through the employment of two knowledge engineers. And at Least one of these individuals should be experienced. One knowledge engineers should be given the primary responsibility for conducting the interview, while the other knowledge engineer listens carefully to both the questions and responses. The second knowledge engineer will also make sure that the meeting is being properly recorder and, when necessary, replace tapes and move microphones.

At Least one of the knowledge engineers should be experienced in knowledge acquisition and the successful of development of expert systems. One of the worst mistakes being made in expert system developments is the assignment of inexperienced personnel to the effort.

One must always keep in mind that the purpose of these meetings is to extract the knowledge base of the expert. While this sounds obvious, it has been observed that the discussions in some knowledge acquisition meetings meander off onto tangents as the discussants pursue pints that have Little if any beaming on the knowledge base.

There are several modes through which the knowledge base may be extracted during such meeting. One is to simply ask the expert system to explain the procedure through which he or she arrives at a conclusion. Another approach is to conduct demonstration sessions wherein the expert is asked to precede through the decision-making processor a series of examples. In general, the second approach lends itself better to knowledge acquisition.

One of the practices that we employed with considerable success is to ask the domain expert to go through a demonstration of the decision making process at our office. We have found that is an excellent way to determine just what data the domains expert actually requires decision making.

At the conclusion of each session, the knowledge engineers will typically try to restate the responses of the expert in the production rule format. Thus, after a few

sessions, it should be possible to develop a simple, prototype expert system. Once prototype is available, we may use it to extract additional knowledge from the expert.

There are a number of good papers that discuss the conduct of the knowledge acquisition phase. In particular, we have provided some excrement guidelines for knowledge acquisition. Here, we have attempted to summarize our thoughts on knowledge acquisition, where the influence of numerous other' authors is acknowledged. These guidelines are presented in the List that follows.

4.7.1 Selection of the Domain

- The domain should be one for which the expert systems approach is truly appropriate, and for which expert system would provide some distinct advantage over any alterative methods.
- Good decision-making within the domain should be of sufficient importance to management that they are wining to commit the time and resources necessary to support the development and implementation of expert system.
- Management must recognize both the costs and risks of expert systems development knowledge engineers, over a reasonable period of time.
- The domain should be relatively stable; in particular, dramatic changes over the period of the development effort should not be foreseen.

4. 7 .2 Selection 0f the Knowledge Engineers

Ideally, two knowledge engineers should be used, where at least one of these is experienced in development and implementation (successful) expert systems. The knowledge engineers should not be one thick pony. That is, they should at the very I east be aware of alternative approaches to decision analysis.

The primary skins of the knowledge engineers should be in the areas of eliciting knowledge and forming model of that knowledge.

4. 7 .3 Selection Of the Expert

- Ask the organization to provide you with the names of candidate domain experts, that is, those individuals who are believed to have significant expertise within the domain in question.
- Select a domain expert whose performance is generally acknowledged to be above and beyond that the most others performing the task.
- Select an expert with successful track record over a period of time.

- Select experts who is both wining and able to communicate personal knowledge, and who relatively articulate in doing so,
- Select an expert who is both willing and able to devote the time necessary to support the development effort.
- If no expert can be identified, or made available, consider the development of the rule base through alternative means.

4. 7 .4 The Initial Meeting

- Prior this meeting, the knowledge engineers should make an all-out effort to familiarize themselves with the problem, the domain, and the terminology used within the domain.
- Locate this meeting in comfortable surroundings.
- This meeting should be conducted in a informal, relaxed manner.
- Tell the expert what your plans and goals are, and explain just what an expert system is and what it can do (cannot do) for the expert as well as the organization.
- Explain the evaluating of the expert system.
- Reinforce your discussion of expert systems with the damnation of the use of some existing exert system. However, avoid the demonstrations of an expert system that is all too obviously a toy.
- If audiovisual recording is desired, ask the expert for permission to do so -and explains that these recordings will before the private use of the knowledge engineering team.

4.8. Organization of Follow on Meetings

- Attempt to minimize the possibility of interruptions. Set aside meeting times during which the expert can devote his or her full attention to the effort.
- Establish a formal agenda for each meeting.
- Establish goals and objectives for each meeting.
- Once prototype expert system has been developed; establish access to the supporting software and hardware.

4.9. Conduct of the Follow on Meetings

- Elicit the roles through discussion and demonstration.
- Attempt to identify all external sources of data and information that are used by the expert.

- Be patient. Don't interrupt the domain expert.
- A void criticism -instead, focus on clarification.
- Always remember that you are building a model of the expert's role base, not a model of your role base.
- If you don't understand a point made by the expert, don't be afraid to admit it. Ask for clarification.
- Use test cases to both demonstrate the decision making process and identify the Limits over which the role based is valid.
- Acquaint the domain expert with production roles; this may encourage the expert to being stating his or her roles in this format.
- Always remember what you are there for.

4.10. Documentation

- Document the results of the meeting as soon as possible after the meeting (preferably, immediately after the meeting)
- Documentation for each meeting should include such facts as:

  -Date, time, and Location for meeting.

  -Name of expert.

  -List and description of the rules identified during the meeting.

  -List of any new objects, attributes and/or values encountered

  -Their properties.

  -Identification of any new outside sources and references.

  -Listing of new terminology encountered, and associated definitions.

  -Listing and discussion of any gaps or discrepancies encountered.

  -Remainders.

- Documentation in support of all production rules thus far developed should include such facts as:

  -A Listing and description of all rules thus far developed.

  -A listing and description of all objects, attributes, and values thus far encountered.

  -Source and reference List

  -Glossary of domain terminology

  -Listing and discussion of the test cases used to evaluate the prototype.

4.11. Multiple Domain Expert Systems

One additional consideration in knowledge based development: the existence of more than one domain expert. Some authors have noted that this situation can be particularly frustrating if not properly and delicately handled. However, he advises that the knowledge engineer need not be particularly consumed about multiple experts. That is, using a rule base cloned from one expert, we build a prototype expert system and then yet the other domain experts critique results.

Our experiences in dealing with multiple experts have followed similar approach. We have always selected on domain expert as the individual from whom the rules were to be acquired, that is, as the key expert. We have presented the prototypes to the remaining experts for a critique. In doing this, we have tried to discourage the key expert from attending such presentations. We feel that his or her attendance may cause the other experts to feel less free making their comments and criticisms.

There is yet another situation in which multiple experts may be encountered. However, rather than having mastery across the entire domain of interest, these experts may each have expertise in various portions of the domain. One approach to this situation is to develop a set of expert systems, one for each sub domain. Another is to utilize separate knowledge basis and to coordinate these through single expert systems package by means of the black boarding approach. And this precisely the approach used in HEARSAY.

5. UNCERTAINTY OF EXPERT SYSTEMS

5.1. Source of Uncertainty

Before examining methods for dealing with uncertainty in expert systems, one should first appreciate precisely just and why uncertainty exists. In general, there are two primary sources of uncertainty that may be encountered in an expert system:

- Uncertainty with regard to the validity of a knowledge base rule
- Uncertainty with regard to the validity of a user response

Consider first the uncertainty associated with a rule. For example, consider the heuristic rule given by "if a dog barks, then it will not bite" (from the old saying, "a barking dog won't bite"). Whoever the canine expert was who came up with this rule may either believe it is always true, or may simply believe that it is true in general. In the first case we could assing a confidence factor of, say, 1 to the rule, that is, it is true 100 persent of the time. In the second case, we need to assing some value less than 1. Of course, the question is, "What should that value be?" If we believe that 8 times out of 10 the rule will hold, we might then assing a confidence factor of 0.8 to the rule, and the confidence factor would thus reflect a subjective estimate of the probability of the validity of the rule. If we have no confidence whatsoever in the rule, we might assign a value of 0 as its confidence factor.It is rather unlikely that we would be so naive as to believe that any confidence factor assigned in so subjective a manner is absolutely accurate. Rather, the use of such confidence factors tends to simply indicate, on a strictly relative basis, our confidence (or lack of confidence) in a rule. And we simply cannot take confidence factors too literally.

The second source of uncertainty is associated with the response, or responses, of the user of the expert system; spesifically the replies provided in response to generated user queries. Consider, for example, the following query to the user (in this case, a physician) of a medical diagnostic expert system:

```
Does the patient have severe stomach cramps?
```

Where the expected answer is evidently either "yes" or "no". Now, if the patient is doubled over in agony, the physician would most likely answer in the affirmative.

However, what if the pain in the patient's stomach is not quite so extreme? A strictly "yes" or "no" response to such a question may be unsatisfactory. We might then ask the user to reply with a confidence factor. For example, we might(arbitrarily) use a scale of 0 to 10 where a 0 represents a judgmentthat there is no basis upon which to assume that there is a stomach disorder while a 10 indicates that the patient is experiencing the most intense pain imaginable. We might then rephrase our query as

```
Indicate the intensity of any stomach cramps (0/10).
```

If the user responds with, say, a value of 8, then we might interpret this as an indication of stomach cramps at a very intense level. Again, however, note that we cannot take the response too literally. Two different physicians may well provide two quite different responses, even when dealing with the same patient and data set.
Now, before we proceed further, note that there is an alternative user prompt in this situation that may be just as good, or possibly even better, than the use of (explicit) confidence factors. That is, we might present the user with the following query:

```
Indicate the level of intensity of stomach cramps:
1. Extreme
2. Very intense
3. Moderate
4. Minimal
5. None
Input response:_____
```

Here, rather than attempting to deal with a numeric value, the user needs only select the response that seems most appropriate. Now, we are not implying thatthe menu prompt is necessarily either better or more accurate than the use of confidence factors. However, in some cases it may prove more efficient and much more natural.

Returning to the thrust of our discussion, we should note that confidence factors must generally be considered to be subjective estimates of the relative level of confidence one should have in either a rule or user response. They are not necessarily probabilities, and are thus not subject to the rules of probability theory. This is a disturbing fact for many analysts who would like to employ a scientifically sound

approach to the determination of uncertainty in expert systems. When one considers that expert systems are, by definition, made up of heuristic rules, it seems rather absurd to attempt to attach any absolute level of precision to the combination of such rules. However, to begin the discussion of various approaches to uncertainty in expert systems, we shall consider the use of Bayesian probability. And this discussion should more graphically illustrate the complexity of the problem faced.

5.2.Uncertainty Through Baesian Probability

In order to clarify the discussion, let us focus on an expert system for the diagnosis of disk drives for personal computers. To keep things simple, let us assume that there are but two possible outcomes, or conclusions, to the consultation session. That is, the disk drive will either be found to be defective (i.e., in need of repair) or it will be considered fine. Let us further assume that we have been introduced to Mr. Mack N. Tosh, an expert in the diagnosis and repair of personal computer systems. Mack has informed us that, in his expert opinion, if a disk drive is making unusual noises, 8 times out of 10 it , is defective.

In essence, what Mack has done has been to pose a production rule plus a confidence factor (which we shall denote as cf) for the validity of the rule. We might then write this rule as

Rule 1: If unusual noises= yes

Then disk drive status = defective (cf = 0.8)

Such a rule has two very important implications. First, it implies that the user is able to distringuish between normal and unusual noises, a feat that may well be beyond the capabilities of the layperson. Second, and if confidence factors are actually probabilities, this rule implies that the probability of the alternate conclusion (i.e., the disk drive is good) is 1-0.8, or 20 percent. That is, if probability theory holds in such situations, there should be a companion rule such as the one given below:

Rule 2: If unusual noises = yes

Then disk drive status = good (cf = 0.2)

However, and perhaps not so surprising, studies have shown that many experts will reject such an implied rule. Thus it seems evident that human experts are simply not using probability theory, at least as we know it, as the basis for their estimates of certainty.

Let us now turn to the use of empirical data in conjunction with Bayesian probabilities. Bayes' formula may be used to determine the probability of a given conclusion (c) given certain evidence, or facts (f). The formula is given as:

$$P(C\backslash f) = [p(f\backslash c)*p(c)]/p(f) \quad (5.1)$$

Where $p(f) = p(f/c)*p(c)+p(f/\sim c)*p(-c)$ (5.2)

And $P(c\backslash f)$ = probability of conclusion C given facts f

$P(f\backslash c)$ = probability of facts f given conclusion C

$P(f)$ = unconditional probability of facts f

$P(c)$ = unconditional probability of conclusion C

$P(\sim c)$ = unconditional probability of not conclusion C

$P(f\backslash \sim c)$ = probability of facts f given not C

For our specific example, p(c\f) is the probability of a bad disk drive given unusual noises; p(f\c) is the probability of unusual noises given a bad disk drive (i.e. the drive may be defective and not produce any strange noises); p(f) is the probability that any disk drive is noisy: p(c) is the probability that any disk drive is defective; p(~c) is the probability that any disk drive is not defective; and p(f\~c) is the probability of unusual noises from a good disk drive. Why it is so difficult to employ formal probability theory for the development of confidence in production rules shold now be clearer. Specifically, it is not at all an easy (or, in particular, practical) task to determine empirical (or even good, subjective) values for p(f\c), p(f), p(c), p(~c) and p(f\~c).

However, for the sake of discussion, let us assume that empirical studies have taken place wherein it is noted that, when a computer has a defective disk drive, the emannation of unusual noises will occur in 90 percent of these computers (a value that might, at first, appear to lend some weight to Mack's subjective estimate provided by rule 1, earlier). Thus, we now know (if the empirical studies were performed correctly and if sufficient evidence was available and examined) that p(f\c) = 0.90. Let us also assume that we have enough empirical evidence to determine that disk drives are bad on 2 percent of all computers (i.e., p(c) = 0.02); and thus p(~c) = 0.98. The only information that is still required is p(f\~c), that is, the probability of noises coming from a good disk drive. Assuming once again that we can collect data to determine this, let us presume that noises come from good disk drives about 8 percent of the time (i.e., p(f\~c) = 0.08). We can now enter all of this information into Eq. (5.2) to determine p(f):

$$P(f) = 0.90*0.02+0.08*00.98 = 0.0964$$

Substituting for p (f) in Eq. (7.1) we obtain

$$P(c\backslash f) = [0.90*0.02] / 0.0964 \approx 0.187$$

Conseqently, for our hypothetical example, we have determined that unusual noises indicate a bad disk drive with a probability of about 18.7 percent. Comparing this result with that originally stated by the expert (i.e., Mack), as listed in rule 1 above, we can note a very substantial discrepancy (i.e., a confidence factor for the rule of 80 percent versus 18.7 percent). It has been noted that, in actual practice, such discrepancies are both common and significant.

One might be led to believe that the probabilistic approach is likely to lead to better results, having been developed through scientifically based, quantitative *methods*. However, this would only be true if the probabilities used to compute p(c\f) in Eqs. (5.1) and (5.2) are accurate. In many real-life situations, such accuracy simply does not exist. Even more likely, it may simply be unrealistic to gather the empirical data necessary to compute the probabilities.

In general then, the Bayesian approach is an appealing method for the determination of the certainty of production rules, but it is quite often just not the practical for actual expert systems of any realistic size or complexity. We should also note that the approach described deals only with the confidence factor associated with a rule, that is, it does not explicitly consider the uncertainty associated with user responses.

5.3. "Uncertainty": The EXSYS Aproach

Since this text includes the EXSYS expert systems shell, we will remark briefly upon the approach employed by this package for uncertainty. The demonstration version supplied with this text is based on the standart EXSYS package. Thus ,we shall restrict our remarks to uncertainty as employed in this particular version of EXSYS. The demonstration version permits the use of three different methods for uncertainty. These are

- The 0 to 1 system
- The 0 to 10 system
- The –100 to 100 system

The first method simply assigns a 1 to any final conclusion clause that is true, and a 0 to any one that is false. Note carefully that the only two values permitted are 0

and 1 (i.e., we do not, as in some systems, permit the fractions between 0 and 1). As such, this method is not really intended for dealing with uncertainty as the expert system treats the rule base as deterministic. Consider the following example:

Rule 0-1: If can fly = yes
and caped uniform = yes
and gender = male
then superman = yes-probability = 1
and batman = yes-probability = 0

While we personally feell uncomfortable with calling such factors probabilities, the interpretation of rule 0-1 should be obvious. That is, if the individual sighted can fly, has a caped uniform, and is male , then we are (1) certain tha the is superman, and (2) equally certain that he is not Batman (since Batman cannot fly).

The second method employs all numbers between, and including 0 and 10. Given that we have a final conclusion clause that is reached by a number of rules, the confidence factors (or probabilities, as they are termed in EXSYS) are averaged .There is , however , an exception to the averaging process. İf any confidence factor is either a 0 or a 10, then that value is locked in, and no averaging takes place.

To illustrade the 0-10 method , consider a rule base that has concluded that the user's investment strategy should be

- **Bonds :** with confidence values of 1,5, and 6 (i.e, four different rules, each with a different confidence factor,have been fired and conclude that the invesment strategy should be bonds)
- **Blue chip stocks:** with confidence values of 0,7,9 , and 9 (i.e.i four different rules have concluded blue stocks)

As a result, we average the values associated with bonds to derive a final confidence level of 4 for the investment in bonds. In the case of blue chip stocks, the final confidence value is 0 since the value of 0 from one rule serves to lock in that confidence factor.

The third approach uses all numbers between, and including, - 100 and 100. Here, however, the values of -100, 0, or 100 do not lock in the confidence factor value. Further, you have a choice between simply averaging the results or combining them as either dependent or independent probabilities.

Before proceeding further, we should explain the use of negative confidence factors. EXSYS uses a range from - 100 to 100. Other packages might use -10 to 10, or –1 to

1. Whatever the range ,all are attempting to capture the same thing . That is, in the -100 to 100 system, we interpret.

- - 100 to mean that the conclusion is absolutely wrong, or false
- 0 to mean that we have no confidence in the conclusion
- 100 to mean that we are absolutely sure that the conclusion is true

The use of negative confidence values may be valuable when capturing the rules of a domain expert who reasons from negative results. For example, a physician may note , whenever a certain diagnostic test comes back negative, we can absolutely rule out an entire category of diseases. Or, a safety engineer may use negative reasoning to exclude certain accident causes (e.g., if the expert checks for evidence of human failure, and finds that such a failure may absolutely be ruled out, then he or she may pursue purely mechanical causes).

5.4. Uncertainty Through Fuzzy Sets

Both previous approaches to productions-rule uncertainty focused on the determination of the level of confidence to place in a given rule. They did not, however, attempt to address the source of uncertainty as a consequence of user input. Fuzzy set theory, or fuzzy logic, has been proposed as one means for handling such a situation. Since there is neither the time nor need to provide a complete background on fuzzy sets (those who are interest in such details are directed to the references (Kickkert, 1978; Zateh, 1965; Zimmermann, 1985), we shall simply note that the concept of fuzzy sets was developed by Lotfi Zadeh (1965) as an approach to deal with certain types of problems where a simple yes or no response is inadequate.

For example, consider the question about the height of a person. That is, suppose you are asked whether or not a given individual is tall. If the individual in question is an adult male and is 7 feet tall then the response is clearly affirmative. And if the subject is but 3 feet 6 inches tall, the response would clearly be negative. However, what if the person is 5 feet 10 inches in height? On this case the response is not nearly so obvious. Zadeh proposed that one use a fuzzy membership function, with values over the continuous range of 0 to 1 for such situations. A value of 1 indicates, for our example, that the person is most definitely tall (specifically, about as tall as we would ever expert a person to be). However, some number between 0 and 1 could represent a partial degree of tallness. For example, a man who is 5 feet 10 inches in

height might have a fuzzy membership value of 0.75.

In conjunction with the fuzzy membership function, a of rules for evaluating a conclusion using fuzzy logic has been provided. Specifically,

- If premises are connected by the logical AND operator, we use the minimum of the fuzzy values associated with the premises to determine the composite value for the premises.
- If premises are connected by the logical OR operator, we use the maximum of the fuzzy values associated with the premises to determine the composite value for the premises.
- If the fuzzy value of a premise (e.g., premise i) is given as fv(i) then NOT fv(i) = 1- fv(i).

To demonstrate, consider the following production rule:

Rule1: If disk drive noisy= yes (fv=?)
And disk formatting results in bad sectors= yes (fv=?)
Then disk drive status = defective (cf = 0.9)

Note carefully that the confidence factor associated with the rule itself is 0.9 (i.e., we might believe that the rule is correct roughly 9 times out of 10). Further, if the user provides responses concerning whether the disk drive noisy or if formatting results in bad sectors, he or she must indicate an associated fuzzy value (fv) for each input, where this value lies between 0 and 1. .

Since the disk drive of our hypothetical example is assumed to be quite noisy, let us further assume that the user responds to the prompt for the first premise with a fuzzy value of 0.8. However, since disk formatting only occasionally results in bad sectors (i.e., unusable portions of the disk), the value for the user's response for the second premise is assumed to be 0.3. Using the properties of fuzzy calculus as listed earlier, we would then take the minimum of 0.8 and 0.3, resulting in a composite rule premise confidence level of 0.3. Further, if we have a confidence factor of 0.9 in the validity of this rule then our confidence in the rule's conclusion (i.e., the disk drive is defective) is simply the product of this factor times the fuzzy value of the composite premise, or 0.9*0.3= 0.27. That is the value 0.27 has been obtained by multiplying the composite rule premise fuzzy value (0.3) by rule confidence factor (0.9).

Suppose that we also have rule 2, listed below.

Rule 2: If disk drive chatters when a disk is inserted = yes (fv =?)

Or drive chatters when a disk is ejected = yes (fv =?)

Then disk drive status = defective (cf = 0.7)

Assuming that the fuzzy value for the first premise of rule 2 is 0.9 and that of the second is 0.2, we now take the maximum of these (since the two clauses are connected by the Or operator) to determine the composite rule premise confidence level. Under the assumption that a rule 2 has a confidence factor of 0.7 the confidence in the conclusion of rule 2 is 0.9*0.7 = 0.63.

Note that we now have two different values of confidence in support of the same conclusion clause AV pair, that is, the conclusion that the disk drive status is defective. From rule 1, our confidence is 0.27 and from rule 2 it is 0.63. Specifically, the results of two different rules serve to support the same conclusion. So, what confidence should we now place in the final conclusion concerning the status of the disk drive? One approach is to simply OR the results, that is, take the maximum of the confidence levels of the supporting rules. Consequently, we would then say that our confidence that the disk drive is defective is 0.63.

To summarize, the fuzzy set approach permits the user to respond to prompts with something other than a simple yes or no answer. That is, it allows for partial truths. While the scientific basis for its employment in expert systems is debatable (as is the basis for most alternative approaches), it does provide a straightforward, appealing approach for the subjective inclusion of uncertainty. However, one criticism of the approach (at least in the simple form presented here) is that it fails to consider the amount of supporting evidence.

Consider, for example, two rules that support a specific conclusion (i.e., the same AV pair). If the confidence factor for one rule is 0.2 and the other is 0.5, then our confidence in the conclusion is 0.5. What if, however, there are ten rules that support this conclusion, with confidence factors of 0.2 for the first nine rules and a value of 0.5 for the remaining rule? Our composite confidence in the conclusion remains 0.5, and this seems counterintuitive.

5.5. Confidence Factor Union Method

As a consequence of the result described in the previous paragraph, we may wish to employ an alternative approach to the determination of the confidence of an AV pair, given support by several rules. This method may be termed the confidence-factor union method as it employs an approach analogous to the determination of the union of sets. To illustrate, assume that two rules conclude the same AV pair and union of sets. To illustrate, assume that two rules conclude the same AV pair and that one has a *confidence factor of 0.2 while the other has a confidence factor of 0.5, that is,*

$$Cf_1 = 0.2$$

$$Cf_2 = 0.5$$

Where cf_i = the confidence factor of supporting rule i. Letting C(cf) represent the confidence factor in final conclusion, we have

$C(cf) = cf_1+cf_2-cf_1*cf_2$

Thus, for our problem, we may compute the final confidence factor for the AV pair as

$$C(cf) = 0,2+0,5-0,2*0,5 =0,6$$

This nation may be extended to any number of supporting rules by simply dial with two rules at a time. For example, if we had three rules supporting the same conclusion with confidence factors of 0.2, 0.5, and 0.5, we combine the result obtained directly above (i.e., 0.6) with the confidence factor of the third rule, or

$$C(cf) = 0,5+0,5-0,5*0,6=0,8$$

Obviously, with the confidence-factor union method, the confidence factor for a given AV pair increases with an increases in the number of rules concluding that AV pair. This may seem more intuitively appealing than simply taking the maximum value of the individual confidence factors. However it is still a heuristic approach to the blending of confidence factors.

5.5. A Widely Employed Approach to Uncertainty

The MYCIN project has received extensive exposure and, as a result, has served to influence subsequent expert systems software development. The manner in which MYCIN deals with uncertainty has, in particular, been widely copied. In this section, we

shall present an approach to uncertainty that is based upon that used by MYCIN.

A prerequisite of the method represented here is that the use of the logical OR operator (i.e., in connecting premise clauses) is precluded. This is no problem as we should recall that any rule with an OR operator may be the composed into two simpler rules. Thus, the only considered rules where, if multiple promise clauses exist, they are connected by the AND operator.

We shall use confidence factors ranging from –1 to 1, where a one represents a rule or response in which one has absolute certainty that it is true and a –1 indicates a rule or response that one believes to be absolutely false. A zero than indicates a lack of confidence in the rule or response. The concept of a threshold level will also be employed. The threshold level will be denoted as δ in this discussion. The threshold level used in MYCIN is 0.2, and we have assumed the same level in the discussion that follows (in most packages using such an approach, this level may be set by the user).

We are now ready to consider the conditions associated with the use of confidence factors in this approach. First, let use consider the computation of the composite rule promise confidence factor. This value is given as

$$RI_k(cf)=\min_i \{P_i(cf)\} \text{ if all } P_i(cf)\geq\delta$$

Or $$\max_i \{P_i(cf)\} \text{ if all } P_i(cf)\leq-\delta$$

Or $$0 \text{ if } |P_i(cf)|<\delta \text{ for any } i$$

Or $$0 \text{ if any two } P_i(cf) \text{ are of opposite sign} \qquad (5.3)$$

Where $RI_k(cf)$ = the composite rule input, or promise confidence factor of rule k

$P_i(cf)$ = the confidence factor for promise clause i

δ = the confidence factor threshold level

Next, consider the confidence factor of the output of any rule. This is given as

$$Cf_k = RI_k(cf) * [R_k(cf)] \qquad (5.4)$$

Where cf_k = the (attenuated) output confidence factor of rule k

$R_k(cf)$= the confidence factor of rule k

$RI_k(cf)$= the composite confidence factor of the promise of rule k

Figure 5.1. Provides an illustration of the procedure. Here we have assumed that the certainly of the rule itself is 0.8. The confidence factor of the composite input of the rule (i.e., the composite certainty across all premises) is given simply as

$$RI_k(cf) = \min_i \{Pi(cf)\} = \min\{0.9, 0.7, 0.5, 0.75\} = 0,5$$

Next, to determine the confidence associated with the rule's conclusion, or output, we simply note that the confidence factor of the rule serves to attenuate the certainty associated with the rule's composite input certainty. Thus,

$$Cf_k = RI_k(cf) * R_k(cf) = 0.5 * 0.8 = 0.4$$

We have thus far examined how to determine the confidence factor for a single rule. However, in the general case we must combine the confidence factors for the conclusions of several supporting rules in order to determine a final confidence factor for the associated AV pair. The method used to compute the final confidence factor of an AV pair, denoted as C(cf), as supported by several rules employees Eq.(5.5), below. Note carefully that only rules whose confidence factors exceed the threshold level are employed in the computations. Future, for clarity in presentation, we have assumed that just two rules support the conclusion under consideration. Extension of this formula two more than two rules should be obvious. Or, from a more pragmatic view, we may simply blend the confidence values two at a time. As should be obvious, this formula represents an extension of the confidence factor union method previously presented, that is, two encompass the use of negative confidence factors and a confidence factor threshold level.

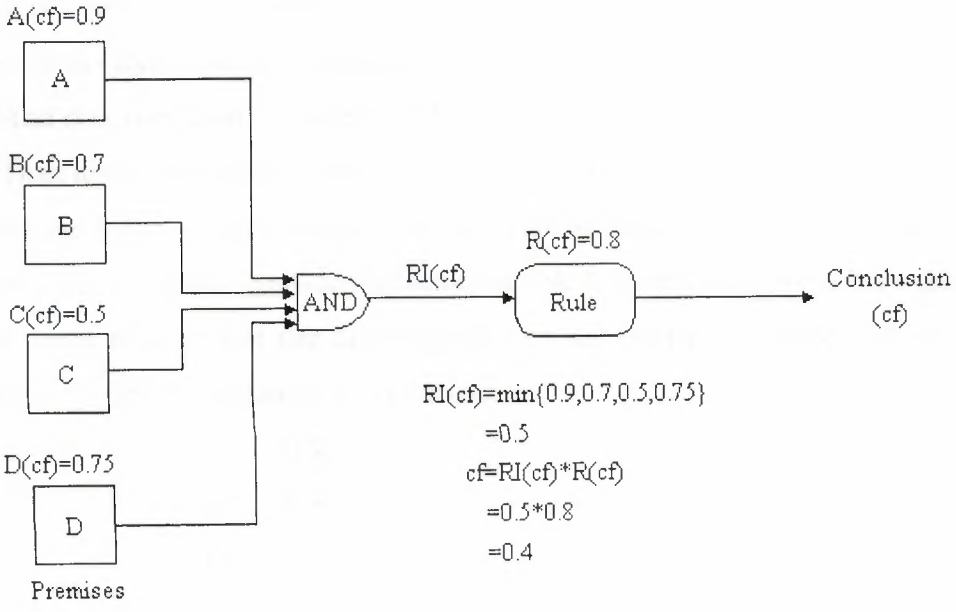


FIGURE 5.5 Confidence in combined premise clauses.

$C(cf) = cf_1 + cf_2 - cf_1 * cf_2$ if cf_1 and $cf_2 \geq 0$

or $cf_1 + cf_2 + cf_1 * cf_2$ if cf_1 and $cf_2 \leq 0$

or $cf_1 + cf_2 / [1 - \min(|cf_1|, |cf_2|)]$ if $-1 < cf_1 * cf_2 < 0$

or 1 if $cf_1 * cf_2 = -1$

Where C(cf)= the confidence factor of the conclusion, C

Cf_1= the confidence factor of supporting rule 1

Cf_2= the confidence factor of supporting rule 2

And both cf1 and cf2 most exceed the threshold level.

Example 5.1: Use of Confidence Factors. To clarify our discussion, let use apply the most recent confidence-factor approach to the knowledge base listed below, where inference will be through the forward chaining approach. Here, we are again attempting to determine if a computer disk drive is defective or not. Please note that we are using this rule base simply for the purpose of demonstration, and no real-world validity be inferred.

Rule1: If drive noise is unusually noisy [$cf_{1,1}$]

And drive age is greater than 1 year [$cf_{1,2}$)

Than drive status is defective [R_1 (cf)=0,8]

Rule2: If disk incretion result is chatter [$cf_{2,1}$]

And screen display is distorted when disk inserted [$cf_{2,2}$]

Than disk drive status is defective [R_2 (cf)=0,7]

Rule3: If disk ejection result is character [$cf_{3,1}$]

And disk formatting is unreliable [$cf_{3,2}$]

Than disk drive status is defective [R_3 (cf)=0,9]

Note that the rule confidence factors [R_k(cf)] are assumed given and that the promise confidence factors (cfk,i=confidence factor for rule k, premise i) are to be supplied by the user upon request. Let use also assume that the confidence factor for the values premises, as ultimately obtained from the user are

$Cf_{1,1}$=0,8

$Cf_{1,2}$=0,5

$Cf_{2,1}$=-1

$Cf_{2,2}$=0,7

$Cf_{3,1}$=0,7

$Cf_{3,2}$=0,3

First examine the propagation of uncertainty through rule 1. From Eq (5.3), we note that the confidence factor for the promise of rule 1 is the minimum of 0.8 and 0.5, or 0.5. The confidence factor of the rule output is than given by Eq.(5.4), the product if the promise confidence factor and the rule confidence factor. This results in a value of 0.4 (i.e., 0.5*0.8=0.4). Next, for rule 2, we note that the promise confidence factors are of opposite sign and thus, from Eq(5.3), the confidence factor of the promise of rule 2 is 0; which, for practical purposes, means that rule 2 any be discarded. From Eqs.(5.3 and 5.4), the confidence factor for the output of rule 3 is simply 0.3*0.9=0.27. As a result, we now have two rules that support our conclusion that the disk drive is defective. Since the confidence factors of the output of both rules are positive, we must employ the first condition of Eq.(5.5).Thus:

$$C(cf)=0.4+0.27-0.4*0.27=0.562$$

That is, the confidence associated with the conclusion, as supported by rules 1 and 3, is 0.562. Note that had we employed the simplified version of fuzzy logic as discussed airliner, our result would have been 4.4.

Let use now try the same example but use the following premise confidence factors:

$$Cf_{1.1}=-0.9$$

$$Cf_{1.2}=-0.5$$

$$Cf_{2.1}=0.1$$

$$Cf_{2.2}=0.15$$

$$Cf_{3.1}=-0.5$$

$$Cf_{3.2}=-0.3$$

In this instance, the confidence factor for the output of rule 1 is –0.5*0.8=-0.4; than confidence factor for rule 2 is 0, and the confidence factor for the output of rule 3 is –0.3*0.9=-0.27. using the second condition of Eq. (5.5) to combine these results, we obtain;

$$C\ (cf)= -0.4-0.27+0.4*0.27=-0.562$$

That is, the certainty associated with a defective this drive is –0.562, or fairly strong evidence that the drive is not defective. And as long as we realize that the confidence factor value derived (i.e., -0.562) is simply an indication of relative certainty, and not an absolute measurement, we may use this bit of information to support our decision-making process.

6. EXPERT SYSTEMS FOR MEDICAL DIAGNOSIS

6.1.Stomach Diseases

In this chapter, we consider the implementation of expert system for medical diagnosis of human illnesses. Firstly we are going to examine diagnosis of stomach disease. To develop expert system for this problem we collect knowledge from experienced specialist and different references. On the base of collected information we have created knowledge base for diagnosis stomach diseases. Knowledge base consists of two parts: diagnosis and recommendation. In diagnosis part, knowledge base has production rules that have premise and conclusion parts. Premise part of expert system includes system inputs. Those inputs are: level of indigestion after eating meal, jack of appetite, pain, laboratory investigations etc. The conclusion part of diagnosis includes the name of diseases (gastritis, stomach ulcer and stomach cancer and their different levels). After defining disease, expert system makes recommendation for its treatment. In this case, knowledge base of premise part includes levels of diseases defined in the result of diagnosis (gastritis, stomach ulcer and stomach cancer). Conclusion part gives recommendation for treatment of diseases. In table 6.1 and 6.2 the knowledge base for diagnosis and recommendation for treatment of stomach diseases are given.

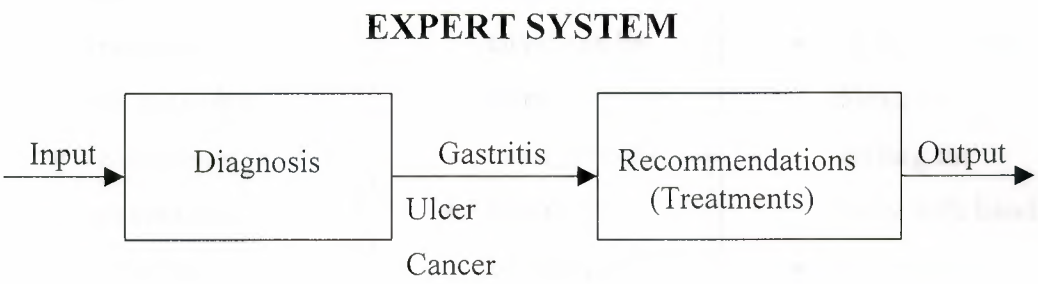


Figure 6.1. Simple diagram of expert system for medical diagnosis

This expert system will behave as a doctor. And it checks inputs according to diagnosis (i.e. according to knowledge) and using this knowledge base, it will determine disease and recommendations for its treatment.

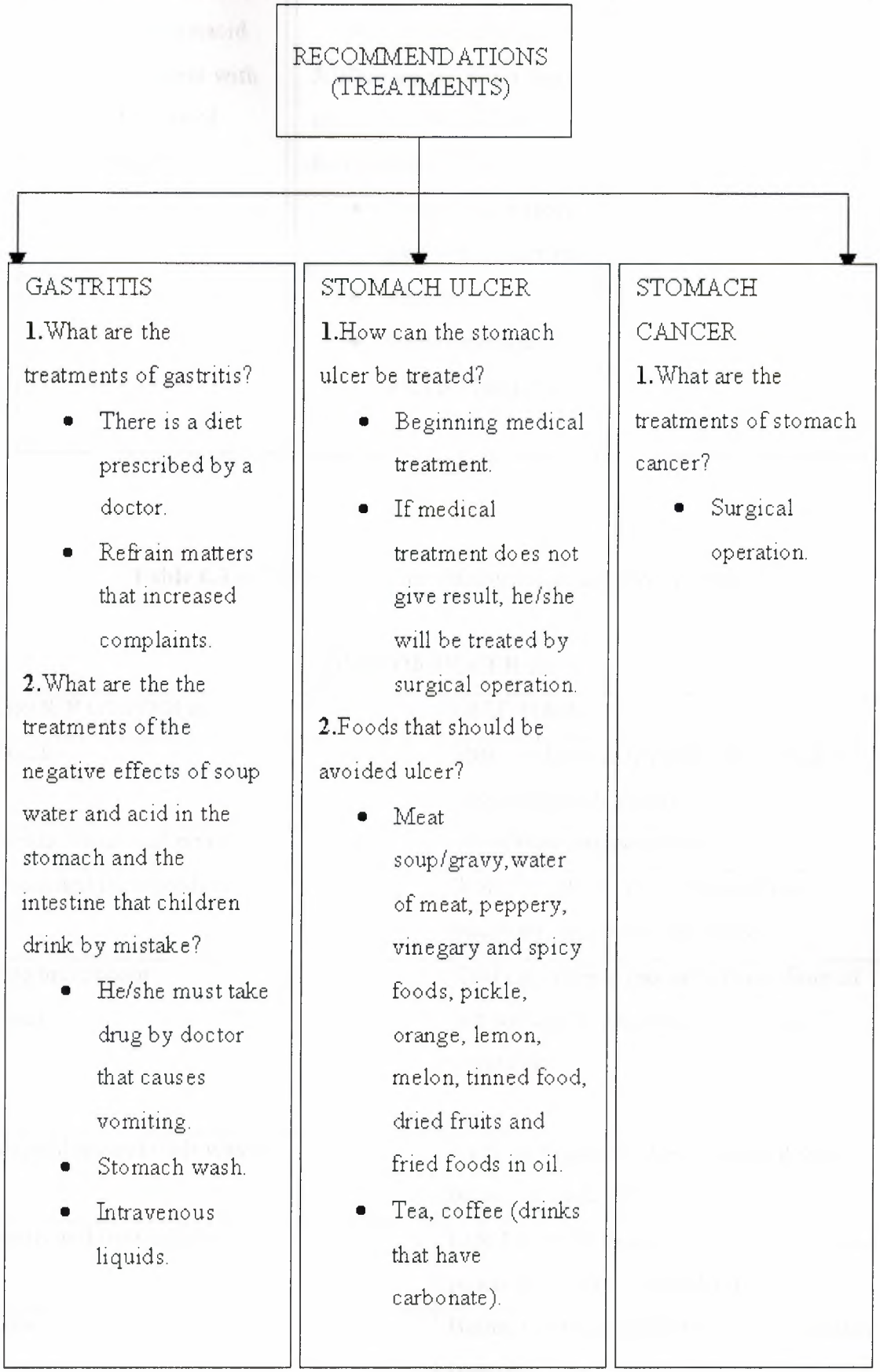
Table 6.1. Main diseases of the stomach and their diagnosis

DIAGNOSIS

GASTRITIS:	STOMACH ULCER	STOMACH CANCER
1-What are the symptoms? • Indigestion after meal. • He/she can have a sensitivity zone on the belly above the navel. • Lack of appetite, fullness upper side of the belly zone, he/she can have a complaint such as ulcer. • Sometimes, nausea and vomiting. **2**-In the laboratory investigations, which results are observed? • Laboratory investigations are usually normal.	**1.**What are the symptoms? • After meal and before taking the meal, pain belly, above navel can be found with the hand pressure. • Sensitivity and stress on the belly, above navel can be found with the hand pressure. • Result of bleeding of the ulcer can be observed anaemia.	**1.**What are the symptoms? • In the belly, weight and swelling. • After swallowing hard foods, there is a discomfort and felling of swelling. • A mass can be found by feeling the belly with hand • There are symptoms such as complaints of stomach ulcer.

• For atropric *gastritis,they can* determine the inadequacy of acid in the water stomach. • For chronic hypertrophic gastritis, there is an action in the stomach. **3.**What are the negative effects of soup water and acid in the stomach and the intestine that children drink by mistake? • It can pierce in the stomach • Difficulty in swallow can be observed. • Intense burning on the zone of the belly. • There is nausea, diarrhea and vomiting. This vomiting is usually bloody. • There is pain like cramp.	• At more *intensity, with* vomiting and relief after vomiting and loss of weights. • Sometimes, pain can spread to the left side of belly.	• Colour of *faeces is deep* black accompanied by anaemia. • Results of vomiting, contents of this come form of coffee grounds.

Table 6.2. Table shows recommendations (treatments) of main disease of the stomach



• He/she can be given antacid treatment with the way of mouth.	• Cigarette • Alcolic drinks **3.**What are the drugs that should not be used by those having ulcer? • Drugs that degrees pain of rheumatism • Aspirin • Corticosteroids and ACTH Phenacetin.	

Table 6.3.a. Types of diet for different stomach ulcer patients

DIET OF ULCER (1)

GROUP OF FOODS	FREE FOODS
Drinks	Milk, linden, garden sage, drink made with sweet and fresh yogurt.
Meats, fishes and poultry	All of these are prohibited
Grain and their products	White bread, biscuits, cracker, rice, macaroni, vermicelli, hardtack.
Egg and cheese Soup	Hard egg, cheese (no salt), floor, floor of pea and lentil, soup made of puree of vegetables
Vegetables and their waters	Puree of vegetables (pea, quash, green beans, spinach, potato, carrot)
Fruits and fruit juice	Ripe banana cooked with peeled apple and peach as form of stewed fruit
Oils	Butter, olive oil, sunflower oil, margarine, corn oil.

Desserts	Sugar, honey, jam (without grain), pudding, rice pudding, cream etc.
Foods (tastes)	Light salt, sauces using milk and flour.

Table 6.3.b. DIET OF ULCER (2)

GROUP OF FOODS	FREE FOODS
Drinks	Milk, linden, garden sage, drink made with sweet and fresh yogurt.
Meats, fishes and poultry	Meat of calf, sheep, lamp, chicken, cattle, turkey and fish (all of these are billed or grilled)
Grain and their products	White bread, biscuits, cracker, macaroni, and vermicelli, simple sugared dried pastry, pie of thin layer of dough, soup of semolina.
Egg and cheese	Soft boiled or hard egg, white cheese, sheep cheese.
Soups	Filtered vegetable soups, flour soup, vermicelli, rice and plateau soups (without water of meat)
Vegetables and their waters	Well cooked carrot, climbing kidney beans, squash, beat purslane, chard, potato and water of carrot and tomato
Fruits and fruit juice	Rip banana, cooked with peeled apple and peach as form of stewed fruit
Oils	Natural oils (butter, olive oil, sunflower, margarine and corn oil that have small amount acid)
Desserts	Sugar, filtered honey, jam, pudding, rice pudding, jelly, pudding made of rice flour and shredded chicken
Foods (tastes)	Salt, milk and flour, sauces made of oil.

Table 6.3.c. DIET OF ULCER (3)

GROUP OF FOODS	FREE FOODS
Drinks	Milk, linden, garden sage, drink made of yogurt, milk with banana, tea, lemonade, milky coffee
Meat, fishes and poultry	Meal of calf, cattle, sheep, chicken, turkey, liver, kidney, spleen that are boiled are grilled
Egg, cheese, grain and their products	All of them are free
Soups	All kind of soups made of cooked in simple water and water of tomato
Vegetables	All of cooked vegetables, water of tomato and carrot
Fruits and fruit juice	All free
Desserts	Simple cake, pudding, rice pudding, cream, honey, jam, stewed fruits, desserts with jelly, grape molasses, dessert of pumpkin
Oils	All free
Foods (tastes)	Salt, creams, all spice, cinnamon, dessert red pepper, thyme, mint and cumin, olive

Table 6.4. The general table for diagnosis

	GASTRITIS		STOMACH ULCER	STOMACH CANCER
	NON-CRONIC	CRONIC		
PAIN	After meal Small	After meal Small	Before and after meal Large	Very large
SENSITIVITY	Small	Middle	Large	Very large
NAUSEA &	Small	Small	Large	Very large

VOMITING	(Small nausea)		(At more intensity)	
BLEEDING	No	Small	Middle (Anemia)	Large

Using the figure 6.2. we created table 6.1. for knowledge base. The aim of this table is to write production rules easily for expert system.

There are 256 combinations. This means there are 256 production rules for expert system. Some of these rules are below:

IF PAIN=SMALL
AND SENSITIVITY=SMALL
AND (NAUSEA_OR_VOMITING)=SMALL OR
(NAUSEA_OR_VOMITING)=SOMETIMES
AND BLEEDING=NO
THEN DISPLAY ("NON-CHRONIC GASTRITIS");

IF PAIN=SMALL
AND SENSITIVITY=SMALL
AND (NAUSEA_OR_VOMITING)=SMALL OR VOMITING=SOMETIMES
AND BLEEDING=SMALL
THEN DISPLAY ("NON-CHRONIC GASTRITIS");

IF PAIN=SMALL
AND SENSITIVITY=SMALL
AND (NAUSEA_OR_VOMITING)=SMALL OR (NAUSEA OR
VOMITING)=SOMETIMES
AND BLEEDING=MIDDLE OR BLEEDING=ANEMIA
THEN DISPLAY ("NON-CHRONIC GASTRITIS");

IF PAIN=SMALL
AND SENSITIVITY=SMALL
AND (NAUSEA_OR_VOMITING)=SMALL OR
(NAUSEA_OR_VOMITING)=SOMETIMES
AND BLEEDING=LARGE

THEN DISPLAY ("NON-CHRONIC GASTRITIS");

IF PAIN=SMALL
AND SENSITIVITY=SMALL
AND (NAUSEA_OR_VOMITING)=SMALL
AND BLEEDING=NO
THEN DISPLAY ("NON-CHRONIC GASTRITIS");

IF PAIN=SMALL
AND SENSITIVITY=SMALL
AND (NAUSEA_OR_VOMITING)=SMALL
AND BLEEDING=SMALL
THEN DISPLAY ("CHRONIC GASTRITIS");

IF PAIN=SMALL
AND SENSITIVITY=SMALL
AND (NAUSEA_OR_VOMITING)=SMALL
AND BLEEDING=MIDDLR OR BLEEDING=ANEMIA
THEN DISPLAY ("CHRONIC GASTRITIS");

IF PAIN=SMALL
AND SENSITIVITY=SMALL
AND (NAUSEA_OR_VOMITING)=SMALL
AND BLEEDING=LARGE
THEN DISPLAY ("CHRONIC GASTRITIS");

IF PAIN=SMALL
AND SENSITIVITY=SMALL
AND (NAUSEA_OR_VOMITING)=LARGE OR (NAUSEA_OR_VOMITING)=AT
MORE HIGH INTENSITY
AND BLEEDING=NO
THEN DISPLAY ("NON-CHRONIC GASTRITIS");

IF PAIN=SMALL

AND SENSITIVITY=SMALL
AND (NAUSEA_OR_VOMITING)=LARGE OR (NAUSEA_OR_VOMITING)=AT
MORE HIGH INTENSITY
AND BLEEDING=SMALL
THEN DISPLAY ("CHRONIC GASTRITIS");

IF PAIN=SMALL
AND SENSITIVITY=SMALL
AND (NAUSEA_OR_VOMITING)=LARGE OR (NAUSEA_OR_VOMITING)=
AT MORE HIGH INTENSITY
AND BLEEDING=MIDDLE OR BLEEDING=ANEMIA
THEN DISPLAY ("ULCER");

IF PAIN=SMALL
AND SENSITIVITY=SMALL
AND (NAUSEA _ OR_VOMITING)=LARGE OR (NAUSEA OR_VOMITING)=
AT MORE HIGH INTENSITY
AND BLEEDING=NO
THEN DISPLAY ("NON-CHRONIC GASTRITIS");

CONCLUSION

In practice some of processes are characterized by hard formalized and unpredictable infom1ation in addition to uncertainty of environment. Analysis of these processes shows that the use of traditional technology for controls these processes leads to non- adequate their description. To solve this problem, the development of an expert system is considered within this thesis.

The architecture of an expert system for medical diagnosis is proposed and the functions of its main blocks are described. The main problem of expert system development is the construction of knowledge base. Using knowledge of experienced specialists and medical references, the knowledge base for the stomach and intestinal diseases is developed. This knowledge base contains more than 1 000 production rules. Premise part of rules includes the main input characteristics of diseases, whereas the conclusion part is the diagnosis and recommendation for the treatment of illness. After defining diagnosis the system provides recommendation for the treatment of illness. The procedure for interpreting the knowledge base rules is developed.

The obtained results satisfy the efficiency of the applied methodology.

REFERENCES

1. AI WEEK," Big Eight Accounting Firm DvevelopsTax ES," AI WEEK, July 1, 1988a,p.4.

2. Rafik Aliyev, Fuad AlÏyev and Mamedgasan Babaev. Fuzzy Process Control and Knowlegge- Engineering in Petf-oe~çal and reb6tiçManufacturing, V-erilag TUV Rheinland, 1991, p-.146.

3. Aikens, J. S., J. C. Kung, E. H Shortl.iffe, and R. J.Fallat, " PUFF: An Expert System for Interpretation otPulmo-nary Function Data, " inB. C.-C-laneey and E. H. Shortliffe, (eds..), redings in Medical ArtificiallntelligenCe: The First De~ade, Addison- Wesley , Readings, Mass., 1984

4. Anderson, D., and C. Ortiz, "AALPS: A Knowledge-Based System for Aircraft Loading," IEEEExpert, Winter 1987, pp-. 71-79.

5. Benchiınol,G.,P. Levine, and J.C. Pomerol, Developing Expert Systems for Business, North OxfordAcademİc, Londo-n, 1987.

6. Bonisso-ne, P. P., and H. E. Johnson, " Expert System for Diesel Electrİc Locomotive Repair,"

7. Brazile, R. P., and K. M. Swİgger, " GATES: An Airlİne Gate Assİgnment and Trackİng Expert System, " IEEE Expert, Suınmer 1988, pp-. 3-3--39.

8. Carter, C., and J. Catlett, " Assessing Credit Card Applications Using Machine Leaming," IEEE Expert, vol. 2, no.3, Fali 1987 , pp; 71- 79.

9. Casey, ;J., "Picking the Right Expert System Applications," AI Expert, September 1989, pp. 44-47. .,

10.Cavalier, T.M., J. P. Ignizİo, and A. L. Soyster, "Dİscriminant Analysİs via Mathematieal. Programming: Ün Certain Problems and Their Causes.," Co-mputers and Operations Research, vol.i6, no...4., 1989, pp. 71-79.

1 1. Feigenbaurn, E. A., Interview in Knowledge-Based Systems: A Step-by-Step Gulde-to Getting Started, Second Annual AI Satellite Symposiuın, Texas Instruınents, 1986.

12. Huntington, D., EXSYS Expert Systerns. Development Package, EXSYS Manual, Albuquerque, NewMexico, t985.

13. Ignizio, J.P., " IdentificatİOn of Incompleteness in Knowledge Bases vİa a Rule Dependency Maxtix, " technica:l paper,Department oflndustrİal Engİneerİng, University of Houston, July 1987b.

14. IgniziQ, J .P.," Attribute- Value Pair Tables and Rule Base Architecture," technical paper, UnİversityofHouston, Houston, Tex., 1988.

15. Lindsay, R.K., B.G., Buchanan, E.A. Feingenbuam, and J. Lederberg, ApplİcationsofArtificİal Intelligence for C~mical Inference: The DENDRAL Project, McGraw-HİIl, New York, 198(}.

16. Pople, H. E.jr., "CANDUCEUS: An Experimental Expert System Fo-r MedİcalDiagnosis," İn P. H. WİnstonandK. A Prendegası (eds.), The AI BusİnessM. 1. T ., Cambridge, Mass., 1984 pp. 67-80.

17. Reddy, D. R., L. D. Erman, R. D. Fennell, and R. B. Neely, " The HEARSAY Speech Understanding System: An Example ofthe Recognition Process," UCAI-3, 1973, pp. 185- 193.

18. Shortlifeı E. H., Computer-Based MedİcalConsultatİons: MYCIN,Elsevier, New York, 1976.