NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

EXPERT SYSTEM FOR MEDICAL DIAGNOSIS

Graduation Project
COM-400

Student: Ebru KÜL (980138)

Supervisor: Assoc. Prof. Dr. Rahib ABİYEV

Nicosia - 2002

ACKNOWLEDGEMENT

First I would like to thank our graduation project supervisor Assoc.Prof.Dr. Rahib ABİYEV, for his patient, kind, well-informed helps.

Then I want to thank Dean of Engineering Faculty Prof.Dr.Fakhreddin MAMEDOV, Chairman of Computer Engineering Assoc.Prof.Dr. Doğan İBRAHİM, my advisors Assists.Prof.Dr. Fa'eq Radwan and Mr.Tayseer AL-SHANABLEH and all the staff of our faculty.

Finally, I want to thank to my mother Naile KÜL and father Dr.Lütfi KÜL for their self-sacrifice, for being my parents. And also thanks to my sisters.And especially I want to thank to my fiance Hamdi SAYIN for his helps and affection.

THANKS....

ABSTRACT

Increasing the complexity of the technology process, 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 accurancy. The effective way to solve this problem is the use of artificial intelligence ideas,such as expert systems.

The aim of our project is development of expert system for medical diagnosis. According to this purpose the state of art understanding of expert system for diagnostic 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 diseases are 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 realised on the base of ESPLAN expert system shell.

TABLE OF CONTENTS

INTRODUCTION

The concept of expert systems originated from research in the field of Artificial Intelligence. Expert system were born when it was realized that there was at least one aspect of intelligence that was not based on reasoning. An expert dealing with a problem in his field often uses very simple reasoning, relying more uppon the knowledge gained from years of experience and training. This insight into the role played by knowledge in the cognitive process encouraged AI researchers to build systems that apply simple reasoning mechanisms to knowledge about a very specific area of expertise. Stanford University is generally given the credit for developing the first "expert system", called DENDRAL, in the early seventies. This system was designed for determining the molecular structure of unknown compounds from their spectroscopic data. Systems such as this soon became the first commercially viable applications of artificial intelligence.

The objective of this project is to investigate the development of expert system for medical diagnosis of stomach diseases. This project consists of introduction, five main chapters and conclusion.

Chapter 1 describes the state of art understanding of expert system for diagnostic problem solving and the main steps for development of diagnostic expert system.

Chapter 2 presents the architecture of diagnostic expert system and the description of functions of its main blocks. The operation principles of expert system, interface between system and knowledge engineer and user are described.

Chapter 3 presents the representation model of knowledge and the development of knowledge based for diagnostic problems. Decsription of knowledge presentation models such as frames, semantic networks, logic predicate, neural networks and production rules are given.

Chapter 4 describes knowledge acquisiton, stages of it, different knowledge in the analysis of knowledge and knowledge acquisition methods.

Chapter 5 presents the practical results of medical expert system application for stomach diseases. The obtained results of expert system application are analised.

Finally, the conclusion section presents the important results obtained with in the project.

CHAPTER ONE:APPLICATION OF EXPERT SYSTEM FOR SOLVING DIAGNOSTIC PROBLEMS

1.1.The Expert System Concept

The expert system is recent addition to circle infonnation 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 respond to questions, asks for clarification, makes recommendations, and generally helps the user in the decision-making process. In effect, working with an expert 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 alternative computers do not.

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 rule of thumb used in an expert system are acquired from a real live 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 a special subject with a view to solving problems or giving advice.

An expert system may completely fulfill 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 concerned 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 analogues 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 inthe design of an electrical circuit. This focus on small domains is more a result of technological limitations then 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 resonable 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 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 the discipline and who ultimately present the alternative solutions to the decision-maker are called decision analysts.

To better undrestand 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 one 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 iit is

extremely doubtful that they are any rigorous approach to assist them substantiating or improving those decisions made. Intuition, experience, and judgement 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 assistancein their decision making. Such means have been achieved through an increased dependence on the use of decision models, and r particularly on quantitive models and analytical methods. Today their reliance of corporations and institutions on such techniques as follows:

- Spresdsheets and databases
- Statistical analysis
- Simulation
- Methods of mathematical optimization

While this formal approach to decision-making has certainly not sub planted the use of intuition, experience, and judgement, 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 qualitive aspects involved in making decisions. However, ultimately both qualitive 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. Unffortunately, data alone are 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 neasures of dispersion or variability.

One fundemental rule data is that, to be value, data must be in he 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 theyu simple in the form in which they were originally collected.We may also consider these data to be our engine data failure database.

Table1.1: 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 systems was to provide information directly, ang 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 informations 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 analytic 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 progamming
- Marginal analysis
- Input-output analysis
- Queuing theory
- Inventory theory
- Project scheduling
- Simulation
- Reliability and quality control
- Forecasting
- Group technology
- Material requirements planning

Asumming that can represent our specific problemusing one or more of such models, the associated methodology may then be used to develop a purposed 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 connection 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 modern analytical methods. Using this definition, a block diagram for a general DSS is depicted in Figure 1.1 below.

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 recomended 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 algoritms for the derivation of the solutions for the particular class of mathematical model under consideration. To more

fully appreciate tha DSS concept, as well as the difference between DSS and expert systems, we need to understand algoritms.

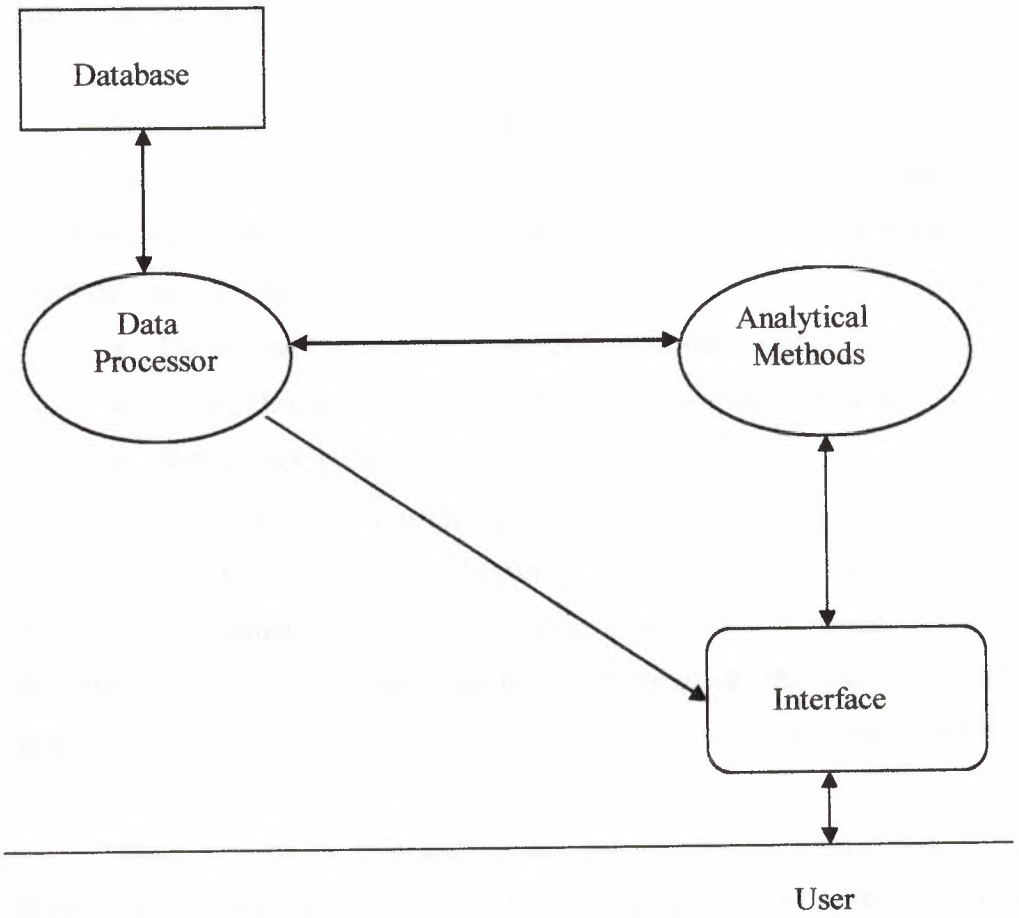


Figure 1.1. A genetic DSS.

1.5.Algorithms and Relationships

One formal definition of an algorithm is "a method for solving a problem using operations from given a 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, division and multiplication. Note most carefully that this definition implies that an algorithm converges.

Algorithms may be applied to an either single mathematical relation 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 representation of our knowledge of how a particular system works.

1.6.Heuristic & Heuristic Programming

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

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

One of the general characteristics of many heuristic is their focus on screening, filtering and pruning. Each of these terms represents just another way to state that heuristics may be used to reduce the number alternatives that are considered. Typically an expert learns through time and experience that certain approaches send 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 theoretically best possible answer.

Not that when one uses heuristics, heuristic programming, and one is implicitly accepting the notation satisfying. Satisfying is a 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 does not guarantee an optimal solution. On designation for the solution procedure is that of an inference process – the procedure that 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 times, give priority to the job that is mostly tardy –or like to become tardy.

Application of those 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 concerned with precisely at the same problem that DSS and 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, to ultimate goal of AI

is to develop an intelligent machine that wills 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 to 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 system 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 limitations. 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 "alternative and conventional computer programming".
- The emphasis, in too much of the expert systems literature, on those factors those 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 learns 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 component engineers-.

- The belief that, just because a person doing a job, he or she is an expert in that job –and 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 System

1.9.1.DENDRAL: An Expert in Chemical Identification

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 In fact, for some tasks, DENDRAL is generally acknowledged as performing better than any human expert. DENDRAL utilizes production rules and was implemented in the LISP programming language. DENDRAL does not have an explanation facility. That is, it simply reaches a conclusion and this conclusion is presented to the user.

1.9.2.HEARSAY I and II: Speech Recognition

HEARSAY-I (1969) and HEARSAY-II (1971) were developed at Carnegie-Mellon University. Specifically, the goal of the system was to have computer understand spoken input. The input to the HEARSAY system is a speech waveform. From this waveform, a set of hypotheses about what may have been said is developed. A best guess flrom this set is then presented as the output.

One of the more innovative concepts developed by the HEARSAY project was that of the use of multiple knowledge bases. One important result of the HEARSAY project was the demonstration that an expert systems approach was superior to what had been the conventional approach to speech recognition. The conventional methodology relied upon statistical tools, that is, the analytical approach. Another result of

HEARSAY was that it spawned several follow-on efforts dealing with the interpretation of several types of signals, in particular, acoustic signals such as those obtained through sonar contacts.

1.9.3.INTERNIST /CADUCEUS: An Expert In Internal Medicine

The goal of INTERNIST (early 1970s) is to perform a diagnosis of the majority of diseases associated with the field of internal medicine. This, in itself, is an ambitious endeavor as there are hundreds of such diseases. However, not only is INTERNIST/CADUCEUS intended to diagnose each disease, it is supposed to consider all the possible combination of diseases that might be present in the patient. It is estimated that the number of such combinations is on the order of 10 to the 40th power. Consequently, as in the case of DENDRAL, we are faced with a problem that exhibits combinatorial explosiveness; a problem for which the heuristic approach is most appropriate.

1.9.4.MYCIN: An Expert in Blood Infections

MYCIN (mid-1970s) is, at this time, probably the most widely known of all expert systems thus far developed. And this is despite the fact that it has never been put into actual practice. The particular role proposed for MYCIN was that of providing assistance to physicians in the diagnosis and treatment of meningitis and bactericidal infections. The knowledge base of MYCIN contains the heuristic rules used by physicians in the identification of certain infections. EMYCIN (for Empty MYCIN) is the name given to MYCIN when this specific knowledge base is removed. In many cases, one may collect a knowledge base associated with a different domain and insert this into EMYCIN, where the result is a new, working expert system.

1.9.5.PUFF: An Expert in Pulmonary Disorders

PUFF was developed in 1979. The purpose of PUFF is to interpret measurements related to respiratory tests and to identify pulmonary disorders. PUFF interfaces directly with the pulmonary test instruments used in such measurements. At the conclusion of the testing, PUFF presents the physician with its interpretation of the measurements, a diagnosis of the illness, and a proposed treatment scheme. PUFF is

viewed as an ordinary piece of laboratory equipment, rather than as an intelligent competitor.

1.9.6.XCON (Rl): An Expert in Computer Configuration

XCON was developed for the configuration of VAX computers at the Digital Equipment Corporation (DEC). A VAX computer may be configured in an enormous number of ways, and DEC attempts to configure each computer according to the specific requirements of each customer. Such a problem might be thought of as a type of loading problem wherein a box is to be loaded with equipment to achieve a specific purpose. Of course, the combinatorial complexity of this problem is enormous and, as such, certain heuristic rules must be employed to reach an acceptable configuration within a reasonable time frame.

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

DELTA/CATS-1 is an expert system developed by the General Electric Company in the early 1980s. The purpose of DELTA/CATS-1 is to assist railroad personnel in the maintenance of General Electric 's diesel-electric locomotives. The development effort began in 1981 and the first field prototype was completed in 1983. Over this period, the number of production rules increased from 45 to 1200. 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 DELTA/CATS-1 is its interface with visual support systems.

The expert systems described in the previous section have become almost legendary in the Artificial Intelligence sector. Some of the more recent expert systems:

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

GATES (1988) is in use (in prototype form) at New York's JFK International Airport. The system is being used by TWA to assist ground controllers in the assignment of gates to arriving and departing 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. Optimization methods had been attempted but were simply unable to cope with the real-time demands of the problem. Thus GATES was developed, using PROLOG, and implemented on a personal computer. The system handles about 100 or more flights a day, have direct access to TWA's database, and can create gate assignments in about 30 seconds. Previously, the human experts had needed between 10 and 15 hours to prepare an assignment, and as much as an hour to modify the assignment each morning.

1.9.9.QMR: Medical Diagnostic Expert System

Using the massive knowledge base first developed for INTERNIST, QMR (quick, medical record) assists physicians in the diagnosis of an illness based upon the patient's symptoms, examination findings, and laboratory tests (1988). QMR, which is resident at the University of Pittsburgh, incorporates over 4000 possible manifestations of diseases and is said to perform at a level comparable to practicing physicians.

1.9.10.FXAA: Foreign Exchange Auditing Assistant

Chemical Bank does $750 billion a year in foreign exchange trading. This involves thousands of transactions a day with the paperwork resulting from such transactions weighing in at about 10 pounds per month. As such, the audit volume is well beyond the capabilities of unassisted human auditors. One particularly important type of audit is that of the recognition of irregular transactions. FXAA (1988) has been developed to provide the necessary auditing assistance. FXAA is a rule-based expert system that has evidently made a major impact within Chemical Bank.

1.9.11.Jonathan's Wave: An Expert in Commodities Trading

A number of firms and individuals have developed expert systems for stock and commodity trading. While it is still too early to assess the success or failure of these programs, they have attracted considerable attention, and customers. Jonathan's Wave is just one of these, developed specifically for commodities futures and commodities futures options trading (1988). Incorporated in the program are the knowledge bases of several approaches to commodities trading. Based upon their suggestions, and their past

performance, the system determines the trades to be made. In this manner, the system acts somewhat as though it were using multiple experts to reach its conclusion.

1.9.12.Insurance Expert ax: An Expert in Tax Planning

Insurance Expert ax (1988) created to assist in the identification of tax planning and accrual issues. Through Insurance Expert ax, companies now have access to the knowledge and tax planning skills from across the country. Insurance Expert ax took more than a year to develop and consists of more than 3000 rules.

1.9.13.HESS: An Expert Scheduler for the Petrochemical Industry

HESS was developed at the University of Houston 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. Their function was to determine what product, or products, to produce, at what time, and through which processors. Their performance was measured against the costs of production; production ruins (i.e., products that do not meet specifications and must be either recycled or downgraded), and lost customer sales. HESS was developed using the EXSYS expert system shell, and through a 12-month effort. HESS, which stands for hybrid expert system scheduler, consists of approximately 400 production rules and runs on an IBM PC or compatible. Not only does HESS accomplish the scheduling task previously done by the human expert, it does so on a much more consistent basis. Savings through the implementation of such a system, at a typical U.S. refinery, have been estimated (by refinery personnel) to be on the order of several million dollars each year. Despite this, the package has yet to be fully implemented as a result of the lack of access to the firm's databases.

1.9.14.An Expert in Poultry Farming

The system analyzes 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 the farmer to any diseases the chickens have, or may get.

1.9.15.DustPro: An Expert in Mine Safety

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. DustPro interfaces with monitoring systems (that monitor methane gas emissions and dust). It runs on a PC and takes about 15 minutes to reach a conclusion. The knowledge base' consists of roughly 200 rules.

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 DOE is to attempt to correctly classify a given document through 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. The knowledge base of this shell contains the rules from the classification guides that determine just how to classify a document (e.g., as confidential, secret, or top secret) and the system is being used to relieve the workload of the people previously assigned to this effort.

1.9.17.CODE CHECK: An Expert in Computer Program Assessment

Abraxas Software (1989) has introduced an expert system for evaluation of C source 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. A code that satisfies such checks is more likely to be maintainable and portable. Abraxas personnel note that 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 that may be simplified. In addition, Code check evaluates the portability of the source code by comparing it with the numerous standards now existing for C programs and identifies any code that will not port between DOS, OS/2, UNIX, VMS and Macintosh.

1.9.18.Expert Systems for Faster, Fast Food Operations

Such systems serve to reduce inventory, speed up service, and even act as training assistants. Wendy's, for example, uses expert systems to plan faster and more efficient delivery of inventory-and plans to expand the package into a more far-reaching decision support system. McDonalds incorporates expert systems in its European operations that incidentally, are entirely PC-driven. Such packages provide valuable, timely assistance to managers who are neither familiar, nor entirely comfortable with the pace of activities in such operations. In a 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.An Evaluation of Problem Types

Although we have presented just a few examples of expert systems, we might note that they are fairly representative 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 faced. Here, the symptoms are the data on machinery performance while the diagnosis involves the identification of defective or failed components. 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 a VAX computer, while HESS attempts to construct a schedule.

1.10.1.Classification & Construction Problems: Definitions

Classification is 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 2 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 Figure1.2: Thus, above figure depicts a scheme in which items 1,2, and 5 are loaded on 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 AI (or expert system):

Class 1: 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 2: Characterized by a need to create a plan or configuration and scheduling. This class coincides to what we have termed as construction problems.

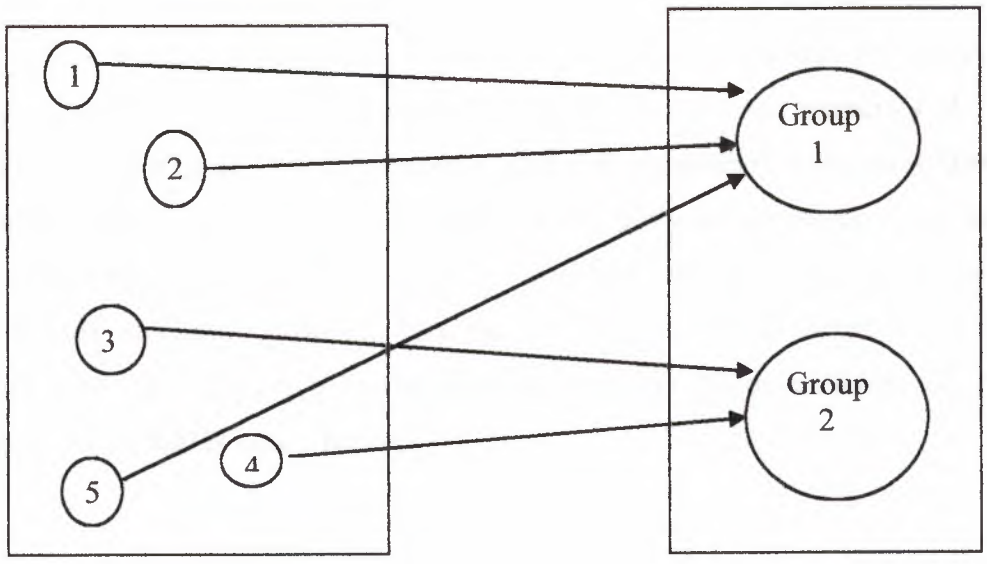


Figure1.2 – Mapping the objects to grouping

Class 3: 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 4: Characterized as applications that humans can handle and computers cannot. Included among this class are such problems as face recognition, reasoning by analogy, and learning how to talk.

1.11.Future Expert Systems

Most of the expert systems that have thus far been discussed in the literature are essentially stand-alone systems. However, in the very near future it is likely that a large portion of the expert systems that are developed will be embedded systems, that is, systems that form only part of the overall software package.

Another form of the embedded expert system is that of the so-called intelligent interface. Intelligent interfaces shall rely, more and more; on expert systems to be better achieve user friendliness in software. Such a system will immediately determine whether or not the user is a novice or expert, and its tailor actions accordingly. That is, the novice user will require more help, support, and guidance, while the more experienced user will need but minimal assistance.

Another trend that we expect to continue is the increased development expert systems-expert systems having 200 or fewer rules. This particular prediction is, however, somewhat, at odds with a commonly held belief of the AI community. A view may have made sense some years ago when expert systems development was somewhat of a trial and error process, and when much of the software for support had to be developed by one. And in a difficult language such as LISP, and with the support of expensive LISP machines. However, with the advent of powerful, inexpensive expert shells-and with implementation on the personal computer-the development of small expert systems are highly cost effective.

CHAPTER TWO: ARCHITECTURE OF EXPERT SYSTEMS

2.1.The Structure of Expert System

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 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 figure2.1.

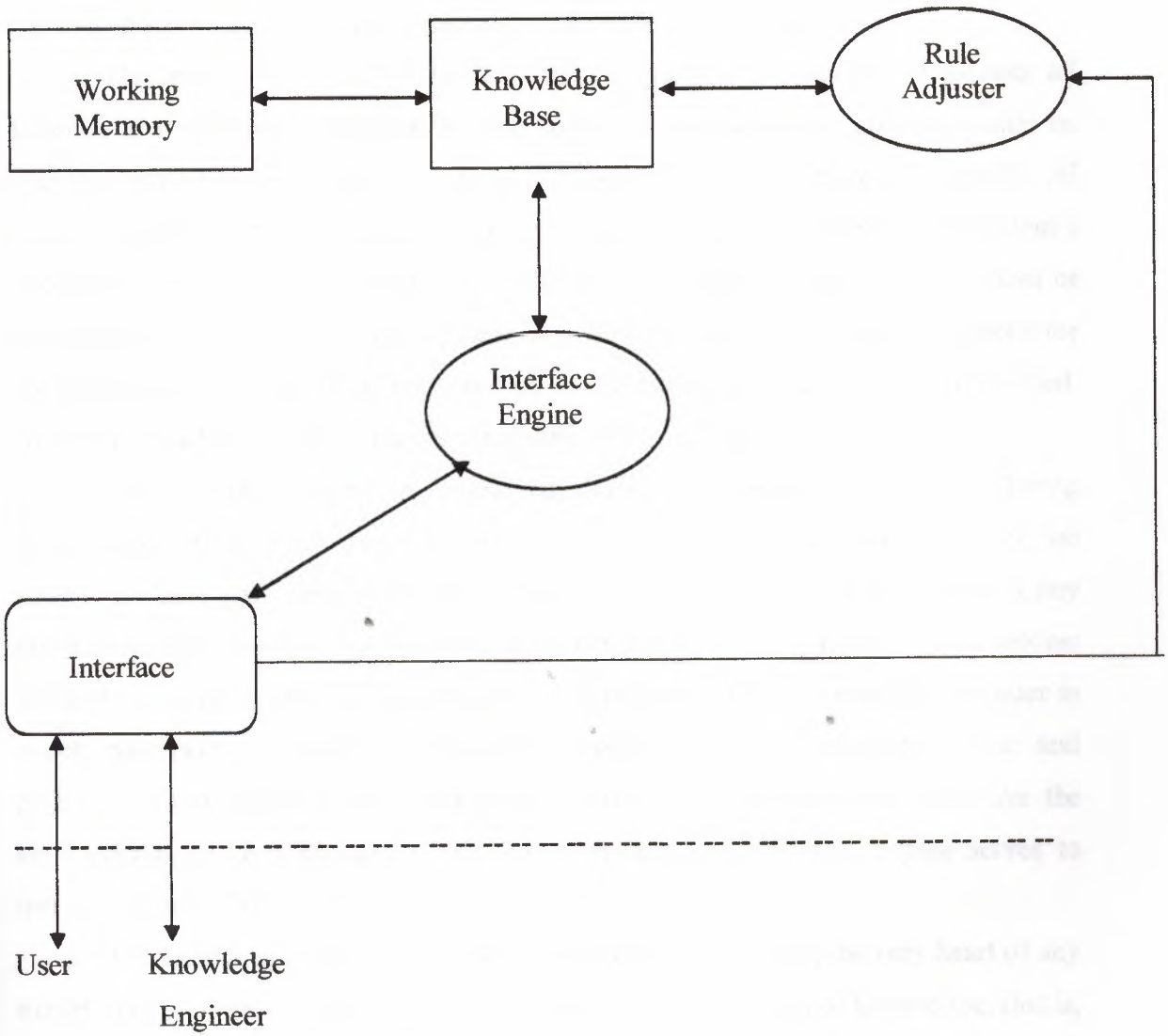


Figure 2.1 –A genetic expert system.

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 for 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 2.1.Heor she accomplishes this through the interface and rule adjuster.

The second type of individual with access to the expert system is designated; in figure 2.1 as simply the user. The 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 outputs. 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 system is that of explanation. That is, some expert system have 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 determined 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 system "shell" includes the entire components listed figure2.1minus 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 figure2.1 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 algorithms and heuristic programming, an expert system separates heuristic rules from the solution procedure. The knowledge base contains a description of "what we know". The inference engine contains a description of "what we do" to actually develop the situation. While the knowledge base changes from domain to domain, the inference engine remains the same.

2.2.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 deduce results in an organized manner. A few techniques for drawing inferences from a knowledge base are described here.

2.2.1. 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** is 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.

2.2.2.Backward Chaining

In this method, one starts with the desired goal, and then attempts to find evidence for providing 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 providing 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. Backward chaining is useful in situations where the quantity of data is potentially very large and where some specific characteristic of the system under consideration is of interest. Typical situations are various problems of diagnosis, such as medical diagnosis or fault - finding in electrical equipment.

CHAPTER THREE:KNOWLEDGE REPRESENTATION

3.1.Models of Knowledge Representation

The knowledge that is contained within an expert system consists of

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

Major concerns deal with how to represent the facts and rules within the knowledge base to:

- Provide a format compatible with the computer.
- Maintain as close as possible a correspondence between this format and the actual facts and rules (i.e., the rules as they are perceived by the domain expert).
- 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.

Several modes of knowledge representation have been proposed. The primary focus will be on rule-based systems for knowledge representation, since it is through this process that the knowledge bases of the expert systems to be described will be developed. However, before we cover rule-based systems in detail, let us first discuss certain alternative modes of knowledge representation.

3.2.Alternative Models of Representation

Knowledge representation modes can be described as:

- OAV (or object-attribute-value) triplets
- Semantic networks
- Frames
- Logic programming
- 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 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 back)

For each attribute, there is an associated value, or set of values. For instance, in the case of the C 130 military cargo aircraft (known as the Hercules), the number of engines is four, the type 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 CI30 aircraft) is never explicitly stated. Actually, the above statements represent AV (attribute-value) pairs. However, associated with any AV pair is some object. Thus, any AV pair implies an OAV triplet.

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. However, rather than 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.2. Here, we note that the C5A is a special type of aircraft (i.e., a large military cargo plane). Further, since the C5A 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 C5A is an airplane, there is no need to store, at the C5A 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.

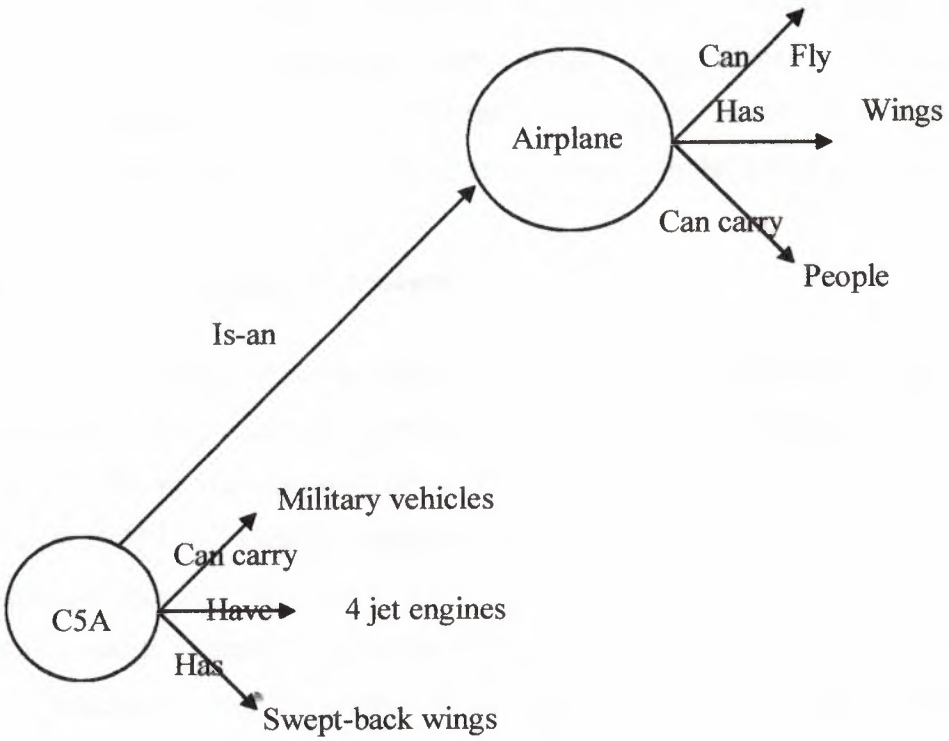


Figure 3.1. Semantic Network

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 for 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.

The primary drawback to the use of frames is, ironically, caused by the very robustness of such a mode of representation. 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.Representation via Logic Statements

The most common form of logic is that known as prepositional logic. A proposition, in turn, is a statement that may be either true or false. Propositions may be linked together with various operators (termed logical connectives) such as AND, OR, NOT, and EQUIVALENT. Linked propositions are termed compound statements.

Predicate calculus represents an extension of prepositional logic. The fundamental elements of predicate calculus are the object and the predicate. A predicate is simply a statement about the object, or a relationship that the object possesses. Predicates may address more than one object and may be combined by use of logical connectives.

One major advantage of the use of logic for knowledge representation is that logic-based languages, such as PROLOG, do exist. However, such languages have been criticized for a certain lack of flexibility a criticism that is becoming less valid with recent enhancements to their procedures. A more immediate and pragmatic drawback of the use of logic for knowledge representation is the fact that one must learn some logic programming language (e.g. PROLOG) in order to develop expert systems.

3.2.5.Neural Networks

Obviously, somehow, some way, the human brain stores knowledge. What is not 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 brain. Specifically, it is thought that knowledge is stored in neurons (or, actually, in the connections between neurons). Figure 3. Depicts a simplified representation of only two neurons within the neural network of the human brain.

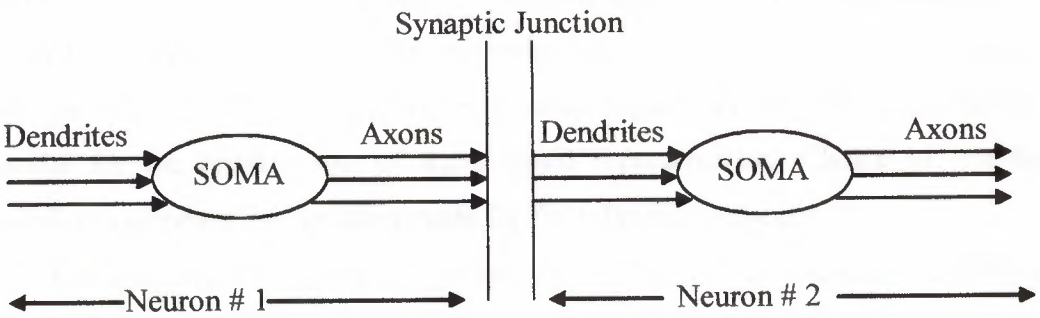


Figure 3.2 A portion of a neural network.

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, impulses are received by the dendrites and transmitted 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. It is believed that the weightings on each neuron-to-neuron interconnection, which in turn influence the level of strength of the interconnecting impulses, might then represent knowledge.

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.

3.3.Representation Via Rule-Based Systems

Undoubtedly, the most popular mode of knowledge representation within expert systems, at least 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 their 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 in particular for the employment of frame-based representation. Choice of rule-based knowledge representation has been made for the following reasons:

- The majority of existing expert system is development packages employ rule bases.
- Rule-based expert systems development packages are normally much less expensive than those employing alternative modes of representation.
- The widespread availability of rule-based expert systems shells permits the knowledge engineer to focus his or her attention on the most critical phase of the development of an expert system, that is, on the knowledge base.
- Rules represent a 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.
- Rule bases can be relatively easily modified. In particular, additions, deletions, and revisior bases are relatively straightforward processes. And this is particularly so in the case designed rule bases.
- Rule-based expert systems can be employed to mimic most features of frame-based representation schemes.

- Validation of the content of rule-based systems is a 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 GRE 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 1a: *If* the student's GRE score is 1350 or more

Then admit the student to the graduate program

Rule 1b: *If* the student's GRE score is less than 1350

Then do not admit the student

For 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 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 a 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 alternate designation for IF- THEN rules is that of condition-action or premise-conclusion statements. The reason for this terminology should be obvious from the above example. That is, given the condition that a student has a GRE of 1350 or better, we take the action of admitting the student to our graduate program. Production rules also contain OAV triplets. Returning again to our previous example and, specifically, to rule 1a, we may note that there is one OAV triplet implied in the IF, or premise, portion of the rule:

Object = student

Attribute = GRE score

Value = 1350 or above

And another OAV triplet is implied in the THEN, or conclusion, portion:

Object = student

Attribute = admission status

Value = yes (i.e., the value to be assigned)

It is particularly important to notice the distinction between the values listed in a rule premise and those listed in a conclusion. In a rule premise, we are testing the value in the statement with any value provided. For example, in the premise to the above rule, we would test the student's actual GRE score value against the value of 1350 (or larger). However, in the conclusion, we are assigning a value to an attribute. Referring again to the above rule, if the student's GRE score is 1350 or above, we may then assign the value of yes to the attribute admission status.

We should also realize that there might be several premise and conclusion statements within a single rule. Each of these is termed clauses (i.e., premise clauses and conclusion clauses). While premise clauses may be connected by 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 termed disjunctive clauses. Premise of a rule, refers to the complete set of premise clauses-in whatever manner they may be connected. The same is true of a reference to the conclusion of a rule, except that in this case the only acceptable statements are either a single conclusion clause or a set of conjunctive clauses.

3.3.2.Attribute- Value Pair Properties

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

Rule 1: *If* grade point average (GPA) equals or exceeds 3.5

Then accept into honor society

In this rule, the attribute for the premise clause is grade point average, and the value to be tested against is 3.5 (or greater). The object has not been specified, but is implied by the rule and/or the particular situation under consideration. That is, the

implied object for the premise clause is a student. The same object just happens to be implied in the conclusion to this rule. When clauses contain only attributes and values, as in the case of the rule under discussion, they are sometimes called attribute-value, or AV, pairs. In the conclusion clause, the attribute-value pair is accept and into honor society. Actually, this is a poor choice of wording for this conclusion. A better conclusion clause might be: ''then, student's acceptance status is yes." While the English may not be quite as natural as before, this restatement permits us to more clearly isolate the attribute and the value. In the restatement, the attribute is clearly student's acceptance status while the value to be assigned is yes. In general, rules should be written so that identification of the attribute and value is straightforward-while the rules remain intelligible. The AV 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 summarized below:

Name: The name of the attribute is simply the wording selected to identify the attribute of the (explicit or implicit) object associated with the clause under consideration. For example, some of the attributes typical of an automobile are; color, number of doors, model, etc,

Type: Attribute values may be either numeric or symbolic. For example, the temperature of a patient may be given in degrees Fahrenheit-a numeric value. Alternatively, we might specify the temperature values to be symbolic, such as high or normal. Yet another set of symbolic values would be yes and 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. Specifically, the only attributes that should normally be provided with prompts are:

- Attributes that appear in a premise statement and never appear in any conclusion statement of the rule set
- 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 values for a person's weight would simply be the set of nonnegative real numbers. If the user 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: Specified values indicate the actual set of values that are either to be tested against (i.e., in a premise clause) or that will be, or have been, assigned (i.e., in a conclusion clause). More specifically, we are concerned with whether or not 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 permit we may deal with uncertainty in either conclusions (i.e., the conclusion attribute values assigned) or premises (i.e., the premise attribute values used). In order to clarify the above discussion, let us examine rule 1, as presented earlier and restated below:

Rule 1: *If* student's GPA exceed 3.5

Then student's acceptance status is yes

Referring to the AV pair of the premise of this rule, the associated properties are

- Name: student's GPA
- Type: numeric
- Prompt: "What is the GPA for this student?"
- Legal values: all numbers from 0 to 4 (i.e., where A = 4.0)
- Specified values: single (i.e., a given student has only a single GPA)
- Confidence factors: none

If we examine the conclusion clause of this rule, the properties of the AV pair are

- Name: student's acceptance status
- Type: symbolic
- Prompt: none (recall that no prompts need be provided for attributes that appear in conclusion clauses)
- Legal values: yes or no
- Specified values: single (i.e., the student is either accepted or rejected)
- Confidence factors: none

3.3.3.Clause Properties

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

- 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 a set of clauses. In the latter instance, we are said to have multiple clauses. Multiple clauses, in turn, may be either conjunctive clauses (each clause connected by the AND operator) or disjunctive (each clause connected by the OR operator). However, recall that disjunctive clauses are not permitted in the conclusion of a rule. Also, note that the premise of a rule may be quite complex.

Another property of a clause is that associated with premise clauses only. This is the property of being either free or specified, and if specified, of being either true or false. If the value of a 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. We shall describe the implications of this distinction in the material to follow. If a clause is not free then such a clause is either true or false .To illustrate these notions, consider the following simple premise clause shown below:

If A = X

Now A must be some attribute for the object about which the clause is concerned. X is then one possible (legal) value for this attribute, and we must test this clause to see if a does indeed equal X. if 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 the clause is true. Otherwise (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 to be straightforward. And indeed they are; however a certain degree of confusion may occur when one employs unknown as an attribute value. Note carefully that we must differentiate between unknown and not yet known. Not yet known means that the value for a respective attribute has not yet been determined (i.e., we have yet to

ask the user for the value-or to attempt to infer a value). Thus, the associated clause is free.

Unknown, however, can be employed in one of two ways. First, it may simply be a legal value for a given attribute. This means is that the premise clause is true, and the rule is triggered. The second manner in which unknown may be employed is slightly more complex, and a function of the specific mode of inference used by the software package. In this case, a value of unknown is assigned to an attribute whenever its value cannot be 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 include

Name: Each rule should have a distinct, as well as descriptive name. Specifically, rather than just labeling a rule by a number or letter (e.g., rule 1, rule A, and so on), it is best to label the rule with a name that serves to concisely describe the contents and/or purpose of that rule.

Premise: Every rule consists of one or more premise clauses. The complete set of premise clauses is termed the rule premise. A rule premise may consist of conjunctive or disjunctive clauses. When we refer to the status of a rule premise, realize that this is a function of the status of the collection of individual premise 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, in turn, two types of rule conclusions: Intermediate conclusions and (final) conclusions. An intermediate conclusion is one that is the conclusion clause of one rule while also serving as a premise clause for another. A (final) conclusion clause is one that does not appear as a premise clause for any other rule.

Notes and reference: 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 specifics of each rule. Many development packages permit the inclusion of notes and references, and this is a feature that should most definitely be employed in any actual knowledge-base development.

(Rule) Confidence factors: When uncertainty is employed, we may associate confidence factors with each rule. The confidence factor of a rule's conclusion is a function of the confidence factors of the rule and the rule premise.

Priority and 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, the 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 in a forward direction-from premise (or facts) to conclusions. In others, 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. In Such instances, we might label rules according to their preferred or default method of chaining, either backward or forward.

Rule status: During consultation, the status of each clause and rule is 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 and associated definitions is provided below:

- The premise of a rule is true whenever a test has been made and it has been determined that the premise has been satisfied.
- 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.
- If the premise of a rule is true then that rule said to be triggered.
- If the premise of a rule is false then that rule may be discarded or, in some cases, made inactive.
- If a rule is fired then this implies that the action implied by the conclusion clause(s) is taken. The values associated with each attribute of the conclusion clauses for this rule have to be assigned.
- A rule that has been fired is no longer active. It is either discarded or, in some cases, made inactive.
- If a rule is to be fired, that rule must first have been triggered.

- If a rule has been neither fired nor discarded, that rule is designated as being active.

3.3.5.Rule Conversion: Disjunctive Clauses

While such conversations 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 Conclusion

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. To 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 $50000
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.

CHAPTER FOUR: KNOWLEDGE ACQUISITION

Definition of knowledge acquisition is the transfer and transformation of potential problem solving the expertise from some knowledge source to a program. Knowledge acquisition is a genetic term, as it is neural with 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 labor 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 will work with, 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.
- **Conceptualization:** 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 casual spatio-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 different data sources.

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

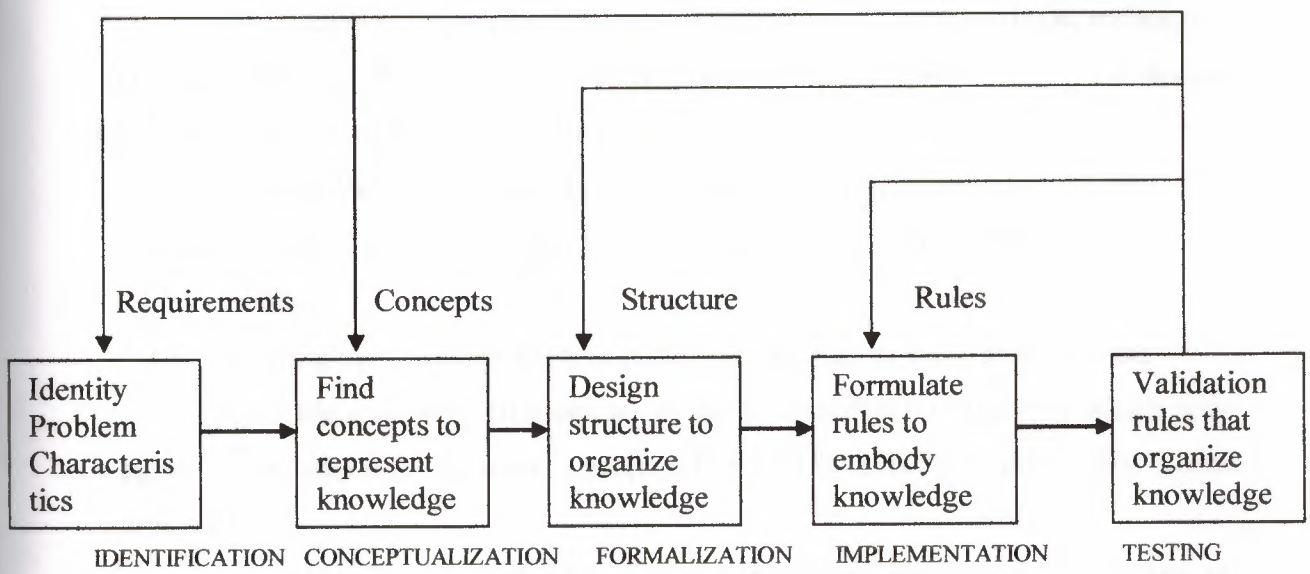


Figure4.1. Stages of knowledge acquisition

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 frames 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 KADS framework is founded on five basic principles, as follows:

1.The introduction of multiple models as a means to cope with the complexity of the knowledge engineering process.

2.The KADS four-layer framework for modeling the required expertise.

3.The reusability of genetic model components as templates supporting top-down knowledge acquisition.

4.The process of differentiating simple models into more complex ones.

5.The importance of structure-preserving transformation of models of expertise into design and implementation.

Today's knowledge engineer is faced with a large space of methods, techniques and tools, which could be used to build an expert system. However, he or she is also faced with three invariant issues, namely:

- Defining the problem that the expert system is meant to solve,
- Defining the function that the expert system will fulfill with respect to that problem,
- Defining the tasks that must be performed in order to fulfill that function.

They made a slightly different set of distinctions level of analysis, which now appear to overlap with the models proposed in KADS-II, bur are nonetheless worth articulating:

- Knowledge conceptualizations aim at the formal description knowledge in terms primitive concepts and conceptual relations.
- Epistemological analysis is concerned to uncover the structural properties of the conceptual knowledge, such as taxonomic relations.
- Logical analysis is concerned with the knowledge about how to perform reasoning tasks in domain.
- Implementation analysis deals with the mechanisms upon which other levels of analysis are based.

4.3.Ontological Analysis

Another knowledge level analysis for expert problem solving is called ontological analysis. This approach describes systems in terms of entities, relations between them, and transformations between entities that occur during the performance of some task. The authors use these main categories for structuring domain 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 transformed 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 eliction 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 fourth. 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 analyze a new set of data.
- Analyze 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.

This elicit-document-test cycle is represented graphically in above figure 4.2.

4.6.Knowledge Based-Knowledge Acquisition

- We shall that many of the learned from trying to expert system technology in various directions have application to knowledge acquisition problem. Specifically,
- Attempt to use expert system as a basis for intelligent tutoring system have led to a deeper understanding of the different kinds of knowledge that experts deploy in problem solving; and
- Attempts to build generic expert system tool like EMYCIN have posed interesting problems concerning how to help developers with the task of encoding knowledge from some arbitrary domain into frame or production rule format.

Such endeavors 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.

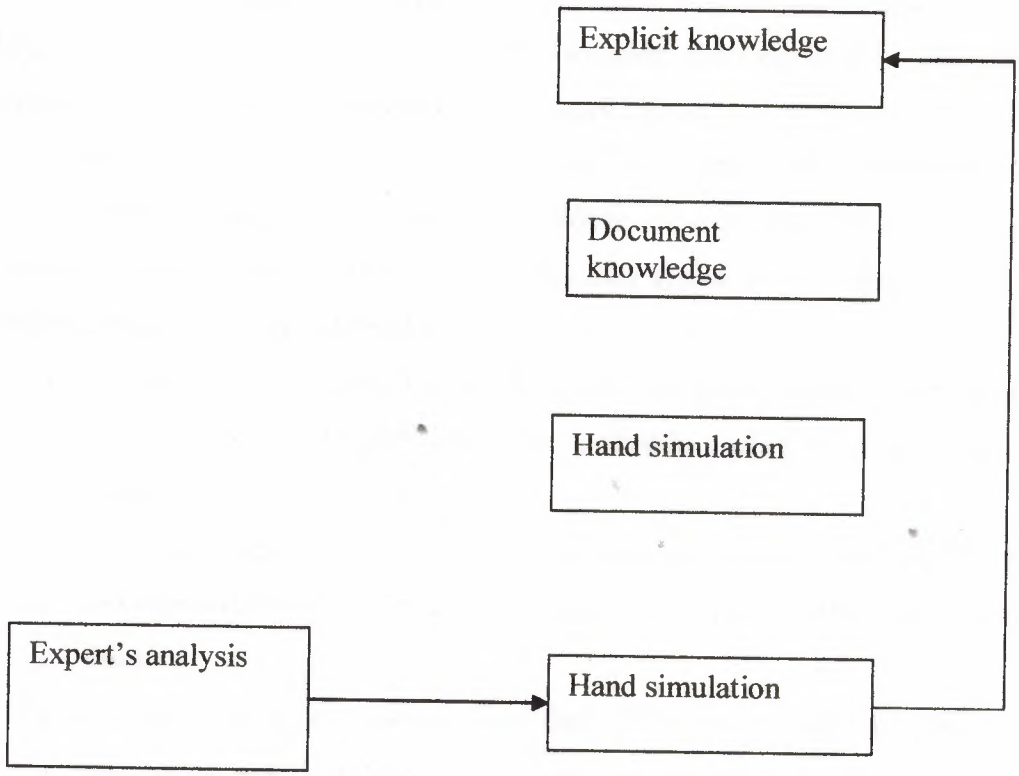


Figure 4.2. Knowledge acquisition cycle in COMPASS

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 elicitation programs needs some knowledge of a domain or a problem area m 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 method that will always be effective.

Knowledge elicitation model by interview based on a domain model is not the last 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 a knowledge base is to go directly to the human expert. However, there are at least four reasons why this may not work, or at least not provide totally satisfactory results.
For some problems, there simply may not be an expert. One example that comes to mind is that 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 actually be exhibiting poor mediocre performance. All too often, the term expert is loosely applied to anyone who simply gets the job done.
- To allege experts may actually be exhibiting poor mediocre performance. All too often, the term expert is loosely 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 meeting should be informal 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 component, we may continue with the knowledge acquisition process. Thus, following initial informal meeting with the domain expert, we should conduct a series of formal meeting 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 engineer 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 points 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 session 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 domain 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 excellent 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 alternative methods.
- Good decision-making within the domain should be of sufficient importance to management that they are willing the 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 of 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 least be aware of alternative approaches to decision analysis.

The primary skills of the knowledge engineers should be in the areas of eliciting knowledge and forming model of text 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 willing 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 an 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 (can not) for the expert as well as the organization.
- Explain the evaluating of the expert system.
- Reinforce your discussion of expert systems with the demonstration of the use of some existing expert system. However, avoid the demonstrations of an expert system that is all too obviously a toy.
- If audio/visual 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.
- 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. Do not interrupt the domain expert.

- Avoid 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 do not understand a point made by the expert, do not 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 concerned about multiple experts. That is, using a rule base cloned from one expert, we build a prototype expert system and then let 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 criticism.

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.

4.12.Knowledge Acqusition Via Rule Induction

An alternative to the acquisition of knowledge through the interface with a human (i.e., an expert or a knowledge engineer assuming the role of the expert) is to convert an existing *(and appropriate)* database into a set of production rules [Carter and Catlett, 1987; Quinlan, 1983, 1987a; Thompson and Thompson, 1986]. The appropriate database, in turn, must consist of data that encompass examples pertaining to the type of problem under consideration— *where the examples selected should represent desirable outcomes* (i.e., it makes little sense to use examples which reflect poor judgment). More specifically, one needs examples of *good* decision making. In some cases, this approach may provide adequate results while, in others, it may at least lead to the development of a credible prototype

system. Several commercial expert systems shells, in fact, incorporate means (actually, supporting programs) for the accomplishment of such a process. We will, in fact, present the results of the application of two commercial software packages (i.e., for the development of rules from a database) later in this chapter. However, one should most definitely first understand precisely how this is done, as well as the scope and limitations of this approach.

Before describing one popular approach to rule generation from data, let us first reflect upon the components of a knowledge base consisting of production rules. To clarify our discussion, let us return to the simplified aircraft identification problem as introduced in Chap. 4. Recall that our problem is concerned with the identification of various types of aircraft, where we will limit the number of aircraft under consideration to just the C130 (Lockheed Hercules), C141 (Lockheed Starlifter), C5A (Lockheed Galaxy), and B747 (Boeing Jumbo Jet). Note carefully that, for purpose of illustration, these four aircraft will be the *only* inhabitants of our universe of airplanes.

4.12.1.Identification of Objects, Attributes, and Values

Rather obviously, the objects in our knowledge base are the four different types of aircraft. Our next step is to consider the attributes that serve to distinguish these aircraft from one another. For example, some of the many attributes exhibited by these aircraft include

- Number of engines
- Type of engines (e.g.,jet or propellor)
- Wing position (i.e.,high on the fuselage or low on the fuselage)
- Wing shape (i.e.,swept back on conventional)
- Tail shape (e.g.,T-shaped or conventional)
- Bulgeson the fuselage (e.g.,aft of the cockpit,aft of the wing,under the wing,or none)
- Size and dimensions
- Color and markings
- Speed and altitude

Having identified a candidate set of attributes and their values, we should next consider filtering out those attributes that do not serve to support our decision-making process. For example, the number of engines is obviously not necessary since each of

the four aircraft under consideration has exactly four engines. (However, in a more realistic aircraft identification model, we would extend our concern to all possible aircraft and there the number of engines does, of course, serve to distinguish one aircraft from another.)

We might also be able to drop the last three attributes since, at the distance we expect to see the aircraft, we will not be able to distinguish colors and markings or estimate size and dimensions, speed and altitude. We are then left with the second through the sixth set of attributes, which may or may not be enough to permit precise aircraft identification. Actually, as we will see, these five attributes (i.e., engine type, wing position, wing shape, tail shape, and bulges) will be more than sufficient—under the rigid assumptions of a strictly deterministic rule base and a completely stable system.

Before we proceed, let us repeat our warning, from Chap. 4, with regard to the avoidance of false economies. Realize that, by eliminating attributes, we are eliminating premise clauses in the rule base, as well as knowledge about the situation. Thus, one must always be careful that the attributes dropped from consideration are not a part of the *necessary* knowledge for the problem under consideration. This guideline can, however, create a dilemma. That is, should we seek the minimal set of attributes or, to be safe, should we try to incorporate every conceivable attribute? Unfortunately, there is no clear-cut answer. Too many attributes may make the knowledge base unwieldy—and require an inordinate amount of data and/or responses from the user. As a result, the expert system will appear to be plodding, if not outright *dumb.* Too few attributes can limit the usefulness of the expert system, as well as make future modifications more difficult. Thus, when all is said and done, we normally seek some acceptable compromise.

Assuming that we feel relatively confident with the filtered set of attributes, we may list these and their associated values. The resulting five attributes are presented in Table 5.1.

4.12.2.The Establishment of a Decision Tree

Recall, from Chap. 4, our discussion of inference networks. These are networks that serve to illustrate the rule base and inference process. Another, and generally more simple network is that of a *decision tree.* It also provides an alternative representation of rule bases (and, as such, is actually another mode of knowledge representation). Such a network may be best explained by means of an example. Consequently, we shall

construct a decision tree for the aircraft identification problem of Table 5.1. Each node in the decision tree will represent either a question about the value of an attribute, or a conclusion. Each branch that emanates

TABLE 5.1

Aircraft attribute listing

	Aircraft Type			
Attribute	C130	C141	C5A	B747
Engine type	Prop	Jet	Jet	Jet
Wing position	High	High	High	Low
Wing shape	Conventional	Swept-back	Swept-back	Swept-back
Tail	Conventional	T-tail	T-tail	Conventional
Bulges	Under wings	Aft wings	None	Aft cockpit

from a node pertaining to a question will represent one of the possible values of the associated attribute. Nodes pertaining to questions will be represented by boxes, while those depicting conclusions will be represented by circles.

Now, for our problem, let us arbitrarily use the node *engine type* as the *root node* of the tree, that is, the node at the highest level in the tree and which includes, under it, all of the possible solutions (i.e., conclusions) contained within the tree. At the next levels in the tree let us use the attributes *wing shape,* then *wing position,* then *tail shape,* and finally *bulges.* Our corresponding tree is then shown in Fig. 5.1. Notice, from Fig. 5.1, that our conclusions are shown at various levels in the tree. For example, as soon as we determine that the engine type is propellor, then we know (at least for our limited model) that the plane must be a C130. Also, we can theoretically arrive at a conclusion wherein we simply do not know what type of aircraft has been sighted. Such events are noted by the question marks as, for example, shown beneath and to the left of the tail shape attribute node. That is, the question mark refers to the fact that, to reach this event, we must have sighted a plane with jet engines, swept-back wings, high wing position, and a conventional tail shape. In our limited universe of planes, there is no such type of aircraft.

We may develop an alternative decision tree by simply reordering the nodes *and* eliminating any branches leading to impossible conclusions. One such ordering is demonstrated in Fig. 5.2. Notice in particular that the decision tree of Fig. 5.2

- Requires fewer attributes to arrive at the identification process (i.e., the attributes *wing shape* and *tail shape* are unnecessary)

• Never results in an impossible event (i.e., the question marks of the previous figure)

Generally speaking, we would prefer to develop a tree with as few levels (i.e., attributes) as possible and with all events representing known conclusions. The fewer the number of attributes, the fewer the data and/or user responses that typically will be needed to arrive at a conclusion. As such, it would appear that the second tree is preferred to the first. And this is essentially true as long as we are dealing with deterministic rule bases.

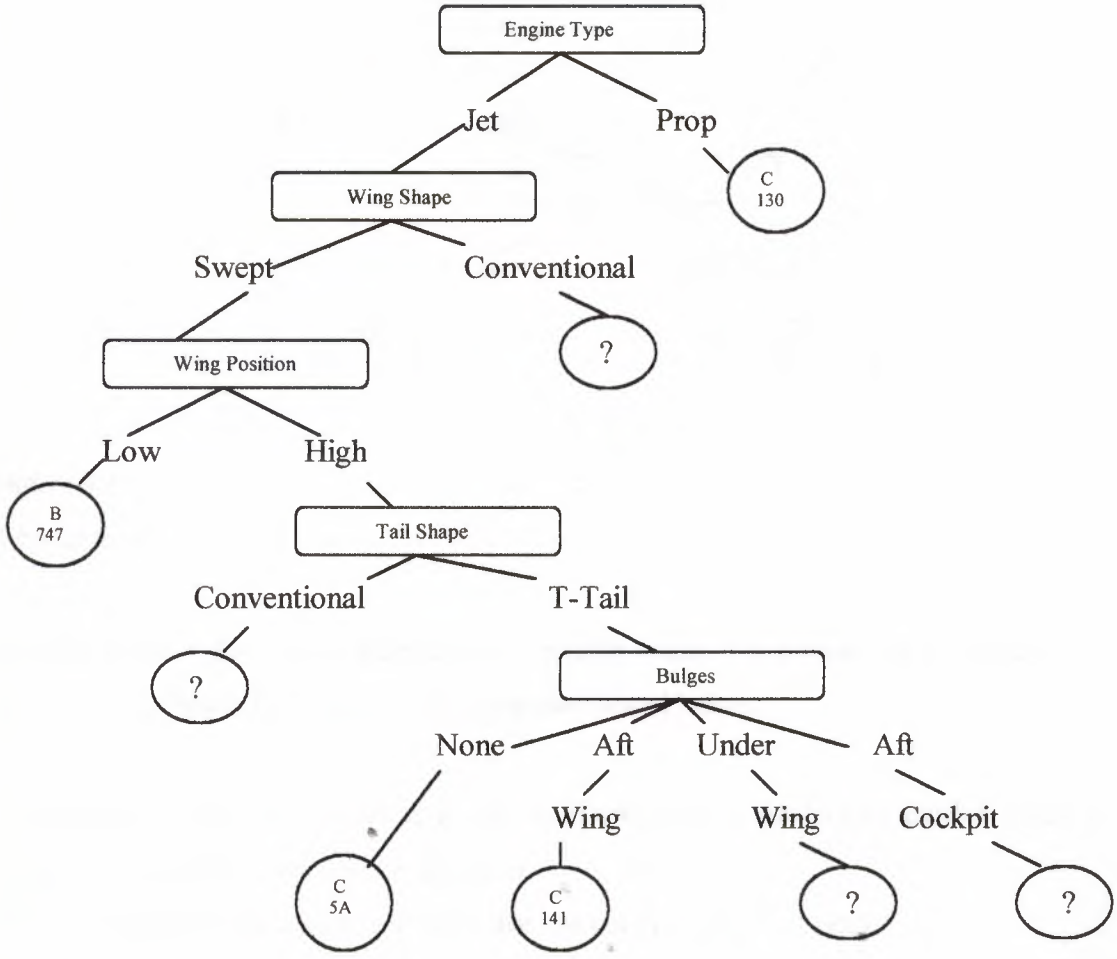


Figure 5.1

Possible decision tree for aircraft identification

4.12.3.Generating Rules from *Trees*

At this point, it should be clear that we can proceed through a systematic aircraft identification process by simply proceeding down the decision tree according to the values of the attributes for the sighted plane. For example, using Fig. 5.2, we may determine that a

sighted aircraft is a B747 by first noting that it has jet engines and then that its wings are positioned low on the fuselage. We can, in fact, do something more interesting, as well as more relevant to rule-based expert systems. Specifically, we can convert the decision tree into a set of production rules. To illustrate this process, we will use the tree of Fig. 5.2. However, first

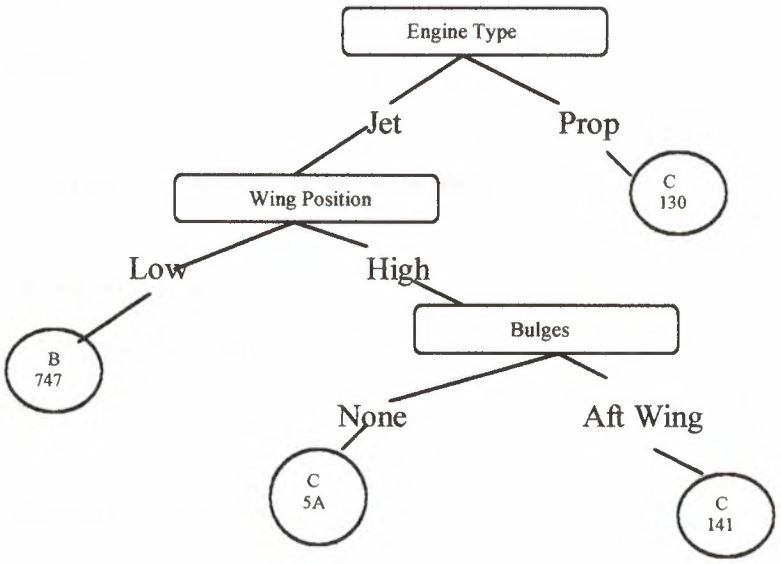


Figure 5.2

A better tree

note that this tree does not contain any intermediate nodes. The absence of such nodes will, in fact, be a prerequisite to the use of the approach listed below:

• *Definition.* A *chain* is defined as a path from one node in the tree to another where we transverse the branches in only one direction.

• *Step one.* Identify any conclusion node that has not yet been dealt with.

• *Step two.* Trace the chain from the conclusion node backward through the tree (i.e., *up* the tree) to the root node (i.e., the highest level node in the tree).

• *Step three.* In the chain, as identified in step 2, circled nodes are considered THEN nodes, or conclusion clauses, while boxed nodes are considered IF nodes, or premise clauses.

• *Step four.* Construct the corresponding production rule set for the chain under consideration, and repeat this process for every conclusion node.

To demonstrate, one conclusion node in Fig. 5.2 is the node concluding that the aircraft is a C130. Tracing back up the tree, we return to the root node which pertains to the question,"What is the engine type?" Consequently, our production rule set for this node is

Rule 1: If engine type is prop
Then plane is C130

Similarly,for the remaining three conclusion nodes,we have:

Rule 2: If engine type is jet
and wing position is low
Then plane is B747

Rule 3: If engine type is jet
and wing position is high
and bulges are none
Then plane is C5A

Rule 4: If engine type is jet
and wing position is high
and bulges are aft of wing
Then plane is C141

And we have, in fact, constructed a knowledge base for the aircraft identification problem that is complete and consistent.

However, before becoming too satisfied with our results, let us consider whether or not our complete and consistent rule set is also efficient. One possible definition for an efficient rule set is one that contains the fewest attributes (and thus, the fewest potential questions). While the rule set constructed from Fig. 5.2 is certainly more efficient than that which would be constructed from Fig. 5.1, it is not as efficient as the one for Fig. 5.3 shown on the next page.

Specifically, the rule set for Figure 5.3 would simply be

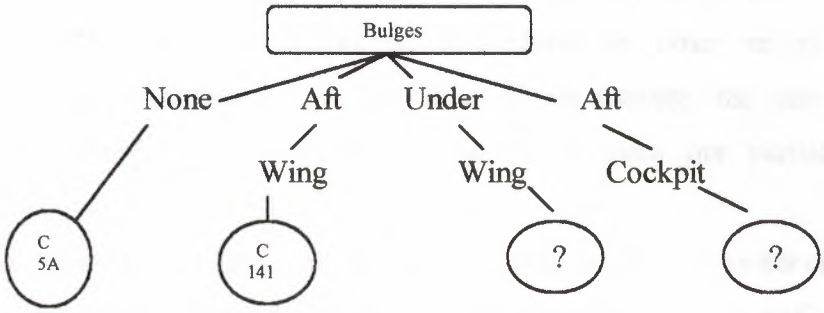


Figure 5.3
The minimal tree

Rule 1: If bulges are none

Then plane is C5A

Rule 2: If bulges are aft of wings

Then plane is C141

Rule 3: If bulges are aft of cockpit

Then plane is B747

Rule 4: If bulges are under wing

Then plane is C130

The only question that needs ever be asked of the user is the one concerning the character of the bulges on the plane's fuselage. For this example, we should by now realize that the most important attribute (the one containing the most information) is that of fuselage bulges. In fact, for this contrived, highly simplified problem, it is the *only* attribute of importance.

Before proceeding further, let us first consider whether or not it is always best to develop a decision tree (and corresponding production rule set) having a minimal number of attributes. If the situation is *deterministic,* and this has been our assumption thus far, then a tree with the minimum number of attributes is, *in theory—and for the static case*—optimal. That is, if an observer is absolutely sure that the aircraft sighted has bulges aft of the wing, and if it is true that the only aircraft in our universe with bulges aft of the wing are C 141s, then we have established the identity of the plane with complete certainty. On the other hand, if the observer is not absolutely positive that he or she saw bulges aft of the

wings (perhaps it was a cloudy day, or the perspective was less than ideal), it could prove extremely beneficial to offer supporting information on other aircraft attributes (e.g., wing position, wing shape). In the case of *uncertainty,* the case most typical of real-world problems, it is generally best to reinforce our partial information.

However,even with a deterministic rule base,it may still prove unwise to minimize the number of attributes.That is,if we design a rule base to minimize the number of attributes (i.e.,premise clauses) for the existing situation,that rule base may have to be extensively modified in the near future-and may require the inclusion of attributes that were not previously necessary. The reader should bear this point in mind in our discussion of various approaches to the development of rule bases from examples.

Returning to our aircraft identification example, we may note that we develop different trees, and thus different production rule sets, depending upon the ordering of the attributes (and, particularly, with respect to the choice of the root node or attribute). In this example, we have generated the decision trees in an unstructured manner. In any real problem, with many attributes and nodes, one would like to have a much more systematic way in which to develop an efficient production rule set. One approach that has been developed to handle this problem is available through Quinlan's ID3 algorithm [Quinlan, 1983]. This algorithm, or variations on the algorithm, may be found in a number of commercial programs. We will now present our version of the ID3 algorithm by means of an illustrative example.

4.12.4.The ID3 Algorithm for Rule Generation

To demonstrate the ID3 algorithm, let us consider a hypothetical problem with regard to the generation of a production rule set for a simplified investment scenario. Specifically, since we may be dubious as to the existence of any true stock market experts, let us attempt to construct an expert system based solely on historical performance. The investment opportunities that we will consider will be limited to

- A mutual fund that invests solely in blue chip stocks
- A mutual fund that invests solely in North American gold mining stocks
- A mutual fund that invests solely in mortgage-related securities

Our goal will be to determine, at any given time and under any given set of conditions, just which *one* of these mutual funds in which to place *all* of our cash holdings. For convenience, let us classify the mutual fund's expected value into three classes: high, medium, or low. These three values then correspond to the conclusions that are to be reached by our expert system.

A much more difficult, and far more critical problem is one associated with the identification of an associated set of attributes. In our aircraft identification example, attribute identification was relatively trivial. Unfortunately, in most problems this factor is far more complex. Further, selection of attributes is primarily an art, coupled with a certain amount of luck, and enhanced by one's insight and experience. For purpose of discussion, we will assume that we have noted that the following factors seem to have the most apparent impact on the values of the three types of mutual funds:

- Interest rates
- Amount of cash available in Japan, Western Europe, and the United States
- The degree of international tension (e.g., prospects for military operations, incidents of terrorism, etc.)

Based on our attribute list, we next peruse historical data to generate the results exhibited in Table 5.2. Each row in Table 5.2 is said to represent a specific example. We will leave, as an open question, just how far back in time one should go in assembling such a data set. Note in particular that the attribute *fund value* represents the conclusion that is sought by the expert system. In order to be consistent with the terminology of the ID3 methodology, we will also term this attribute the *classification,* or *class.* That is, given any scenario (i.e., combination of interest rates, cash availability, and tension), we would wish to classify the mutual fund selections into high, medium, or low expected fund values.

Our next step is to develop a decision tree from this data set. However, rather than employ the arbitrary procedure used in our aircraft identification example, let us use a more systematic approach based on the measure of the entropy of each attribute. Entropy, in turn, is a measure commonly used in information theory. The higher the entropy of an attribute, the more uncertainty there is with respect to its outcomes (or values). Thus, we would wish to select attributes in order of increasing entropy, where the root node of our tree would correspond to the attribute with the lowest entropy value.

The formula for the entropy of any given attribute, A_k, is given as

$$H(C\backslash A_k) = \sum_{j=1}^{M_k} p(a_{k,j}) \cdot \left[- \sum_{i=1}^{N} p(c_i\backslash a_{k,j}) . \log_2 p(c_i\backslash a_{k,j}) \right] \quad (5.1)$$

TABLE 5.2
Investment data set (examples)

Mutual fund-type	Interest rates	Cash available	Tension	Fund value
Blue chip stocks	High	High	Medium	Medium
Blue chip stocks	Low	High	Medium	High
Blue chip stocks	Medium	Low	High	Low
Gold stocks	High	High	Medium	High
Gold stocks	Low	High	Medium	Medium
Gold stocks	Medium	Low	High	Medium
Mortgage-related	High	High	Medium	Low
Mortgage-related	Low	High	Medium	High
Mortgage-related	Medium	Low	High	Low

where

$H(C\backslash A_k)$= entropy of the classification property of attribute A_k

$p(a_{k,j})$ = probability of attribute k being at value j

$p(c_i\backslash a_{k,j})$ = probability that the class value is c_i when attribute k is at its jth value

M_k = total number of values for attribute A_k; j=1,2,…,M_k

N =total number of different classes (or outcomes); i =1,2,…,N

K =total number of attributes; k=1,2,…,K

In table 5.2,we have

- Four attributes (mutual fund-type, interest, cash, and tension); thus $K = 4$
- Three classes (i.e., the fund value is either high, medium, or low); thus $N = 3$
- Three values for the attribute *mutual fund-type* (blue chips, gold, or mortgage); thus $M_1 = 3$

- Three values for the attribute *interest* (high, medium, or low); thus $M_2 = 3$
- Two values for the attribute *cash* (high or low); thus $M_3 = 2$
- Two values for the attribute *tension* (high or medium); thus $M_4 = 2$

and we may compute the values of the entropy for each of our attributes by means of Eq. (5.1). For example, to compute the entropy for the attribute *cash,* we proceed as follows:

$p(a_{3,1})$ =probability that cash is high = 6/9 (i.e., from Table 5.2, cash availability is high 6 out of 9 times)

$p(a_{3,2})$ = probability that cash availability is low = 3/9

$p(c_1 \backslash a_{3,1})$ = probability that a fund value is high when cash is high = 3/6

$p(c_2 \backslash a_{3,1})$ = probability that a fund value is medium when cash is high = 2/6

$p(c_3 \backslash a_{3,1})$= probability that a fund value is low when cash is high = 1/6

$p(c_1 \backslash a_{3,2})$ = probability that a fund value is high when cash is low = 0/3

$p(c_2 \backslash a_{3,2})$ = probability that a fund value is medium when cash is low = 1/3

$p(c_3 \backslash a_{3,2})$= probability that a fund value is low when cash is low = 2/3

and substituting into Eq. (5.1), we have

$$H(C\backslash cash) = (6/9) \cdot [(-3/6) \cdot \log_2(3/6) - (2/6) \cdot \log_2(2/6) - (1/6) \cdot \log_2(1/6)]$$
$$+(3/9) \cdot [(-0/3) \cdot \log_2(0/3) - (1/3) \cdot \log_2(1/3) - (2/3) \cdot \log_2(2/3)] = 1.2787$$

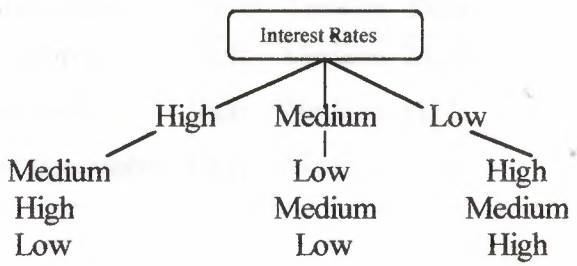


Figure 5.4
Incomplete branching

Similarly, for our remaining attributes, we have

$H(C \backslash interest) = 1.140333$

$H(C\backslash tension) = 1.2787$

$H(C\backslash mutual\ fund\text{-}type) = 1.140333$

There is a tie for the lowest entropy value between *interest* and *mutual fund-type.* Breaking this tie arbitrarily, and constructing a decision tree with a root node as *interest,* we may develop the result shown in Fig. 5.4. Listed below each attribute value in Fig. 5.4 are the classes (i.e., mutual fund values) associated with that particular branching of the decision tree. For example, under *interest* equals *high,* we may note (from Table 5.2) that the associated classes are *medium, high,* and *low.* Since all three branches result in an inconclusive classification, we must branch on yet another attribute.

Since we have already branched on the attribute *interest,* we must partition Table 5.2 according to the attribute *interest.* This results in three subtables (one for interest equals high, one for interest equals medium, and one for interest equals low). The subtable for interest equals high is shown in Table 5.3. Once again, we may compute the entropy for each attribute (i.e., *mutual fund-type, cash,* and *tension),* but now only for the examples provided in the subtable. The result is that the attribute *mutual fund-type* has the lowest entropy (a value of zero). In fact, for all partitions of *interest* (i.e., the other subtables formed when interest equals medium and interest equals low), this will be the case. Thus, we next branch according to *mutual fund-type,* resulting in the tree shown in Fig. 5.5. In this figure, BC, GS, and MR refer to the blue chip fund, the gold stock fund, and the mortgage-related fund, respectively. As may be seen, each branch leads

TABLE 5.3

Classifications when interest equals high

Mutual fund	Cash	Tension	Fund
Blue chip stocks	High	Medium	Medium
Gold stock	High	Medium	High
Mortgage-related	High	Medium	Low

to a unique conclusion (i.e., class) and thus we are finished with the construction of the tree. Specifically, only two attributes are necessary to classify all results.

The associated (and unordered and ungrouped) rule set for Figure 5.5 is then

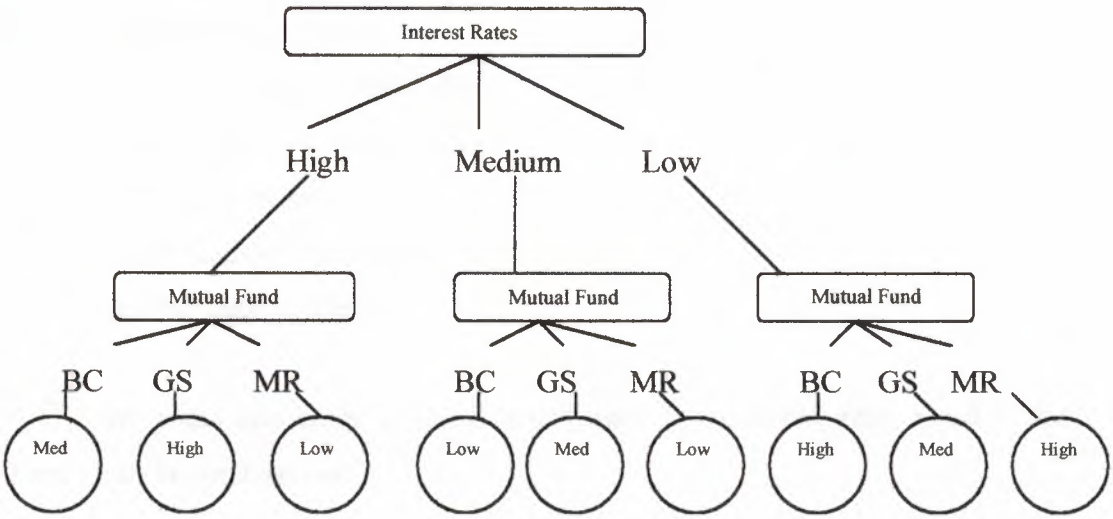


Figure 5.5

Final tree

Rule 1: If interest rates are high
and the fund type is blue chip
Then the value will be medium

Rule 2: If interest rates are high
and the fund type is gold stocks
Then the value will be high

Rule 3: If interest rates are high
and the fund type is mortgage-related
Then the value will be low

Rule 4: If interest rates are medium
and the fund type is blue chip
Then the value will be low

Rule 5: If interest rates are medium
and the fund type is gold stocks
Then the value will be medium

Rule 6: If interest rates are medium
and the fund type is mortgage-related
Then the value will be low

Rule 7: If interest rates are low

and the fund type is blue chips

Then the value will be high

Rule 8: If interest rates are low

and the fund type is gold stocks

Then the value will be medium

Rule 9: If interest rates are low

and the fund type is mortgage-related

Then the value will be high

We could also easily combine certain rules. For example, rules 5 and 8 and rules 3 and 6 may be combined as

Rule 5/8: If fund type is gold stocks

and interest rates are medium

or interest rates are low

Then the value will be medium

Rule 3/6: If the fund type is mortgage-related

and interest rates are high

or interest rates are medium

Then the value will be low

We may, in fact, go even further and, for example, combine rules 7 and 9. However, again recall that the use of disjunctive premise clauses may not be supported by your particular choice of expert systems development package. Further, as more and more rules are joined, the rule base becomes more difficult to read—as well as to maintain.

DON'T GENERALIZE! Now, before we go any further, let us note that the decision tree formed in Fig. 5.5 may lead one to conclude that, at each level in the tree, the same attribute is encountered. For example, at the second level of the tree in Fig. 5.5, we find the attribute mutual fund-type. This, however, is only a coincidence. Typically, there are different attributes at different levels—as well as a different number of attributes in the various chains.

4.12.5.Some Warnings

The conversion of data into production rules is seductively simple, and particularly so in small examples of the type shown above. However, even the above example should indicate some of the problems that are associated with this approach. First, note carefully that while we have nine lines of data (or *examples)* in Table 5.2, they really only represent three distinct data points. That is, we have only listed three possible combinations of interest, cash, and tension scenarios. Thus, the number of examples provided in Table 5.2 is woefully small if we intend to actually construct a reasonably efficient expert system.

We may also encounter conflicts in the database, that is, two examples with identical attribute values may lead to different classes. In such an instance, we most likely have not identified a sufficient number of attributes.

Another difficulty may occur when one deals with attributes having continuous values. If you note our example of Table 5.2, only examples with discrete values (e.g., high, medium, and low) were dealt with. However, it should also be recognized that these values are actually discrete representations of (arbitrary) intervals for continuous values. For example, we noted that interest rates could take on the values of high, medium, or low. In reality, interest rates take on continuous values. Thus, to represent these continuous values by discrete values, we might let

low = interest rates from 0 to 5 percent

medium = interest rates from greater than 5 and up to 9 percent

high = interest rates greater than 9 percent

While this approach certainly eliminates continuous values, it raises yet another question, that is, what ranges of values should one employ for each discrete representation?

The ID3 algorithm will, for problems of any realistic size and complexity, develop very large trees in terms of the number of nodes and branches needed for representation. This results in equally large production rule sets. Unfortunately, large tree sizes do not necessarily result in better expert systems. In fact, it may well be that the effectiveness of the expert system (or classification scheme) is impacted negatively by large trees. Quinlan has developed various tree pruning methods to reduce tree sizes [Quinlan, 1983, 1987a, 1987b; Carter and Catlett, 1987]. One such approach is contained within Quinlan's C4 algorithm (an enhanced version of ID3 that includes *pessimistic* pruning). Using such an approach, a problem requiring 109 attribute nodes and

58 conclusion nodes was reduced to one with but 39 attribute nodes and 17 conclusion nodes. Not only was the size of the tree reduced, but the *hit ratio* (i.e., the proportion of cases correctly identified) was also significantly improved (from about 80 percent for ID3 to over 85 percent for C4).

Yet another problem may occur when one is dealing with stochastic events. Using ID3 for such data may result in trees that are enormously *bushy.* However, Quinlan has also developed extensions of ID3 to at least partially alleviate this situation. Additional comments and criticisms of the ID3 methodology may be found in the references [see, in particular, Mingers, 1986, 1987].

As a further comment on the use of data to generate rule-based expert systems, it should be noted that such an approach is only appropriate when one is faced with a diagnostic type of problem, that is, a problem in which, given certain symptoms (data), we wish to find an appropriate diagnosis (classification). Such a problem is also known by such names as classification analysis, discriminant analysis, or pattern-recognition—and there are a number of conventional approaches to this type of problem. Some, based upon conventional statistical analysis [Johnson and Wichern, 1988; Lachenbruch, 1975], require that the attributes have multivariate normal densities with common covariance matrices (a requirement that is often not met in actual data) while others, based upon mathematical programming methods [Cavalier et al., 1989; Freed and Glover, 198la, 198*lb;* Ignizio, 1987a] are not burdened by such restrictive assumptions. A fair portion of the artificial intelligence community seems, however, to be unaware of the mathematical programming-based approaches to discriminant analysis and all too often justify their use of the ID3 (or equivalent) algorithm primarily (if not solely) on the limitations of statistically based discriminant analysis methods. As a result, a number of rule-based expert systems have been built via approaches such as ID3 when a more appropriate alternative existed.

It may be observed that the ID3 algorithm attempts to minimize the number of attributes in the decision tree. As discussed, this has both good and bad points. One must weigh, based upon each individual case, whether or not the advantages of such an approach outweigh the disadvantages.

Finally, rule induction processes develop what are termed *flat rules.* That is, each rule results in one or *more final* conclusions. This is just another way of saying that rules with intermediate conclusions are not generated. For example, each of the rules of the rule set developed for the examples of Table 5.2 results in a final conclusion (i.e., choice of investment strategy). Intermediate conclusions, however, often serve to document the filtering or pruning

process used by most domain experts to reduce the effort they expend in the search for a conclusion. As an illustration, in order to avoid searching through all stocks listed on a stock exchange, the domain expert may first filter out those that do not pay dividends above a certain amount. This results in a new, reduced set of alternatives. Next, the domain expert may rank the remaining set of candidate stocks according to the price-to-earning ratios (PE ratios), and eliminate those stocks that exceed some certain figure (e.g., a PE ratio of 12). A rule set developed by a knowledge engineer, in conjunction with such an expert should then contain a number of rules with intermediate conclusions—and, as a consequence, the pruning process employed by the domain expert is clearly documented. A rule set developed by means of induction from examples of stock selections will only implicitly include such a process.

4.12.6.Neural Networks and Rule Induction

In Chap. 4, we touched briefly on neural networks as a means of knowledge representation. Recently, neural networks have become a topic of considerable interest within the AI community. They have been found to be particularly effective in dealing with certain types of problems within the general realm of classification (e.g., discriminant analysis and grouping). And one of their primary advantages is that one need not develop production rules, instead neural networks develop (internally) their knowledge through training—on example problems.

However, and as we mentioned earlier, one of the biggest drawbacks of neural networks is that their knowledge base is almost totally opaque. Recently, a fairly extensive effort has been devoted to the induction of rules from neural networks. That is, we may use neural networks to derive the examples that are then presented to rule induction procedures, such as ID3. As a result, a set of rules corresponding to the internal knowledge base of the neural network may be acquired and established—in a transparent mode

4.12.7.Software For Rule Induction

Many commercial expert systems shells provide supporting routines for the development of production rules from examples. This serves to reduce, considerably, the labor involved in such an effort. We will now describe, briefly, the use and evaluation of two particular packages for rule induction.

One of the most popular expert systems shells now on the market is the VP-Expert package available through Paperback Software. All remarks to follow are based on the results

that we obtained when employing versions 1.2 and 2.0 of this package. A somewhat less well-known package (at least in the United States) is Xi Plus, available through Expertech Ltd., out of Berkshire, England. The Xi Plus package used was release 2.0. The evaluation itself was conducted during the summer of 1988.

A discussion of the use of VP-Expert and Xi Plus should help to indicate just how commercial packages employ rule induction systems. In addition, some potentially valuable insight into just how one might approach the evaluation of a proprietary (i.e., closed) software routine might be gained.

The Xi Plus software has a separate routine titled Xi Rule (version 1.00 was employed) that is used to derive rules from data. The method employed by Xi Rule is ID3, our old friend from the previous section.

VP-Expert uses a command termed INDUCE to develop rules from data. However, no explanation is given (at least in the manual that we were provided) concerning the underlying algorithm employed. Evidently, in such an instance, one is expected to accept the methodology employed strictly on faith, an approach that we personally find unacceptable. However, in fairness, it must be noted that VP-Expert is hardly alone in this approach.

There is a real problem in the purchase of software that is probably not discussed in the detail that it deserves. That is, since the source code for the software is usually proprietary (and thus unavailable for examination by the buyer/user), one is left with only three alternatives in regard to the use of such software. First, we might accept the validity of the software on faith. After all, we might think, if the package has sold hundreds or thousands of copies, must it not be all right? Second, we may peruse any existing reviews of the software, or find someone who is using the package and ask his or her opinion. Alternatively, and for the more skeptical, we could develop examples that are useful in exploring the limits of such software, and for detecting problems and inconsistencies. Our personal view, reinforced by experience, is that the third course is the only rational one. Consequently, after receiving copies of VP-Expert and Xi Plus (and Xi Rule), the first thing we did was to test each package on the set of examples provided

TABLE 5.4

Aircraft attribute listing—rule induction data set 1

	Aircraft type			
Attribute	**C130**	**C141**	**C5A**	**B747**
Engine type	Prop	Jet	Jet	Jet
Wing	High	High	High	Low
Wing shape	Conventiona	Swept-back	Swept-back	Swept-back
Tail	Conventiona	T-tail	T-tail	Conventional
Bulges	Under wings	Aft wings	None	Aft cockpit

in Tables 5.4 to 5.7. Although each data set is small and evidently simple, each permits the quick determination of certain features (or lack thereof) of the rule induction software. Such testing is also invaluable, as we will see, in determining the specific mode of operation (and, in particular, of the inference process) of the package.

We have dealt with the aircraft identification example previously and, if you recall, the *only* attribute needed to classify the four aircraft is that of *bulges.* Thus, one might expect that a rule-induction procedure would develop the same set of rules as depicted earlier in Fig. 5.3. The medical diagnosis example is included to determine just how the software deals with examples that are in conflict. Specifically, if the eyes are bloodshot, temperature is high, and nose is runny, then one example states that the disease is a cold while the other states that it is the flu. One would hope that such a conflict, or the potential for such a conflict, would be identified in the rule-induction program. Finally, the investment data set is employed simply to determine if the results we obtained manually using ID3 will be the same as those obtained by the software packages.

The first data set was initially input into VP-Expert. The software includes an editor and screen into which you type the rows of the example data. The rows that should be typed in for the aircraft identification database are shown in Table 5.8. Note that the first row is simply a list of attribute names while the

TABLE 5.5

Aircraft attribute listing— rule induction data set 2

	Aircraft type			
Attribute	**C130**	**C141**	**C5A**	**B747**
Engine type	Prop	Jet	Jet	Jet
Wing	High	High	High	Low
Wing shape	Conventiona	Swept-back	Swept-back	Swept-back
Tail	Conventiona	T-tail	T-tail	Conventional
Bulges	Under wings	Aft wings	None	Aft cockpit
Engines	4	4	4	4

TABLE 5.6

Medical diagnosis—rule induction data set 3

Attribute	**Disease**		
	None	**Cold**	**Flu**
Eyes	Clear	Bloodshot	Bloodshot
Temperature	Normal	High	High
Nose	Clear	Runny	Runny

TABLE 5.7

Investment data—rule induction data set 4

Mutual fund-type	**Interest rates**	**Cash available**	**Tension**	**Fund value *(class)***
Blue chip stocks	High	High	Medium	Medium
Blue chip stocks	Low	High	Medium	High
Blue chip stocks	Medium	Low	High	Low
Gold stocks	High	High	Medium	High
Gold stocks	Low	High	Medium	Medium
Gold stocks	Medium	Low	High	Medium
Mortgage-related	High	High	Medium	Low
Mortgage-related	Low	High	Medium	High
Mortgage-related	Medium	Low	High	Low

TABLE 5.8

Induction table—aircraft identification via VP-Expert

Engine	Wing position	Wing shape	Tail shape	Bulges	Plane
Prop	High	Conv.	CODV.	Under wing	C130
Jet	High	Swept	T-tail	Aft wings	C141
Jet	High	Swept	T-tail	None	C5A
Jet	Low	Swept	Conv.	Aft cockpit	B747

remainder are the set of example data (i.e., as derived by observing the four different aircraft). Further, observe that this table is a *transposition* of Table 5.4.

The rules, as developed by VP-Expert for this example, are listed below:

Rule 0: If engine = prop

and wing position = high

and wing shape = conv

and tail shape = conv

and bulges = under wings

Then plane = C 130

Rule 1: If engine = jet

and wing position = high

and wing shape = swept

and tail shape = T-tail

and bulges = aft wings

Then plane = C141

Rule 2: If engine = jet

and wing position = high

and wing shape = swept

and tail shape = T-tail

and bulges = none

Then plane = C5A

Rule 3: If engine = jet

and wing position = low

and wing shape = swept

and tail shape = conv

and bulges = aft cockpit

and plane = B747

Thus, despite the fact that we only need one attribute (bulges) to classify the four aircraft, the software develops one rule for each aircraft, where the value for *every attribute* must be determined. Rather obviously, VP-Expert is not using ID3.

But, one may ask, is such a rule base *wrong!* The answer is, quite simply, no. Either this rule base or the one employing bulges only will reach the same conclusion. Further, if uncertainty exists (e.g., with regard to the attribute values input to the system), then the rule base developed by VP-Expert might actually be preferred. However, it is still a bit disconcerting to note that, no matter how many attributes are selected, all of these will be employed in the rule bases developed by VP-Expert.

When Xi Rule is applied to the same set of data, we do achieve the set of rules that would be expected, rules employing only a single attribute as shown on the next page:

Rule 1: If bulges are none

Then plane is C5A

Rule 2: If bulges are aft of wings

Then plane is C141

Rule 3: If bulges are aft of cockpit

Then plane is B 747

Rule 4: If bulges are under wing

Then plane is C130

We then modified the aircraft data set to include the attribute *engines* (i.e., number of engines) to correspond to the data set of Table 5.5. For VP-Expert, this is accomplished by simply adding a new column to Table 5.8.

Since there are four engines on all of the planes, one would most likely expect that any package would easily recognize that the attribute engine is unnecessary. Xi Rule once again had no trouble with this example, and the set of rules did not change. However, VP-Expert proceeded to add a new premise clause to each of its four rules: *and* engines = 4. Now, one can argue that this is inefficient or one may argue that we need such knowledge (e.g., in the event that a new plane, with other than four engines, is

added to our universe). While it is much easier said than done, we would have liked to see the rules developed by VP-Expert displayed with those representing unnecessary (or potentially unnecessary) premise clauses highlighted. In this manner, the knowledge engineer can decide which attributes (of those highlighted) to include and which to discard.

Next, we entered the third data set into both VP-Expert and Xi Rule. One would expect the conflict between the examples (i.e., for a diagnosis of cold or flu) to be identified. Xi Rule immediately detected the problem and flashed a warning message on the screen. In addition, the set of rules developed by Xi Rule, as listed immediately below, further served to indicate the existence of the problem:

Rule 1: If nose is clear
Then disease is none

Rule 2: If nose is runny
Then CLASH

Thus, Xi Rule correctly determined that only one attribute is necessary for classification (the value for nose) *and* that there is a potential conflict, or clash, whenever the value of nose is *runny.* However, note that if there really is no conflict (i.e., in the case of multivalued conclusions), then this feature can be more of a nuisance than a help.

When VP-Expert is used to develop the knowledge base from the same set of data, the result is the set of production rules listed on the next page:

Rule 1: If nose = clear
and temperature = normal
and eyes = clear
Then disease = none

Rule 2: If nose = runny
and temperature = high
and eyes = bloodshot
Then disease = cold

Rule 3: If nose = runny
and temperature = high

and eyes = bloodshot

Then disease = flu

When this knowledge base is consulted (i.e., via VP-Expert), it will provide the following answers:

- nose = clear, temperature = normal, eyes = clear: disease = none
- nose = runny, temperature = high, eyes = bloodshot: disease = cold

Thus, even though rule 3 should be triggered (by the latter set of attribute values), it is not fired. Again, there is nothing intrinsically wrong with this result. Rather, it simply tells us something about the mode of control in the inference engine of VP-Expert. Evidently, the first rule that is fired for a given conclusion clause attribute will terminate the inference procedure. However, VP-Expert does have a command statement (PLURAL) that may be employed whenever such situations are encountered. Using PLURAL, multiple conclusions may be reached.

Finally, both packages were tested with the fourth data set. VP-Expert continued to produce rules that included all of the attributes, even though *cash* and *tension* are unnecessary. However, in this instance, it did note the *frequent occurrence* (this is the terminology employed in VP-Expert, not our terminology) of both cash equals high and tension equals medium, and asked if we wished to combine these two into a single occurrence. That is, if one examines the fourth set of data, it may be noted that whenever cash equals high, tension equals medium. (Oddly enough, it failed to notice the *frequent occurrence* of both cash equals low and tension equals high.)

When Xi Rule was applied to this last example, precisely the same tree (i.e., Fig. 5.5) and associated set of production rules were obtained as had been developed by hand in the previous section. This is as expected since Xi Rule employs the ID3 algorithm.

At the conclusion of these four tests, it was clear that the methodologies employed by the two software packages were based on two totally different philosophies. Personally, we would like to see a package that combines both approaches, that is, that (as discussed earlier) lists all attributes but highlights those that are not necessary for the minimal decision tree. In any event, it was only through such an evaluation that the scope, limits, and operating philosophies of the two approaches (and particularly that of VP-Expert) could be ascertained.

4.12.8.Summary

Knowledge acquisition is that phase of expert systems development dedicated to the identification of the rules and facts that comprise the knowledge base. In some cases, such acquisition may be accomplished through interviews with human experts. In others, human experts either do not exist or are unavailable. In this latter instance, one may either attempt to be one's own expert or utilize, if possible, historical data to construct a set of production rules. Yet another alternative is that we might consider training the domain expert to at least recognize the existence of problems that might be approached by expert systems.

Knowledge acquisition through a human expert is a delicate task that needs to be well thought out and carefully and deliberately conducted. Guidelines to this approach exist as a result of the observations of those who have used such a procedure in earlier efforts. However, such guidelines are obviously subjective and incomplete. One can only truly appreciate this task through actual experience.

Acting as one's own expert is generally frowned upon. However, in certain instances there is no other reasonable choice. Further, successful expert systems have been developed (at least in part) through such an approach and thus it is an alternative that should, at least, be considered.

Another alternative to knowledge acquisition may, in some cases, be achieved through the conversion of historical data (or examples) into production rules. The IDS algorithm, and its various extensions, provides one means for accomplishing this when faced with a problem of classification. However, there may also exist more conventional alternatives [Cavalier et al., 1989; Freed and Glover, 1981a, 19816; Ignizio, 19870; James, 1985; Johnson and Wichern, 1988; Lachenbruch, 1975] to such problems that should also be considered prior to making a decision as to the approach to be used.

CHAPTER FIVE: EXPERT SYSTEMS FOR MEDICAL DIAGNOSIS

5.1.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 5.1 and 5.2 the knowledge base for diagnosis and recommendation for treatment of stomach diseases are given.

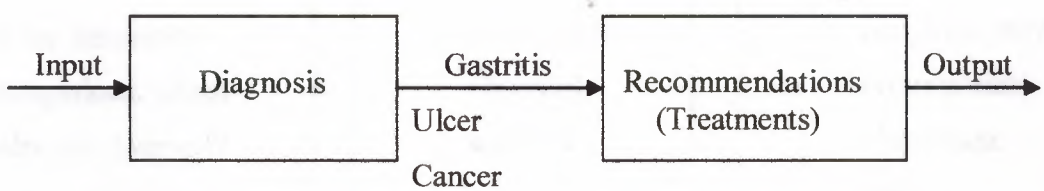
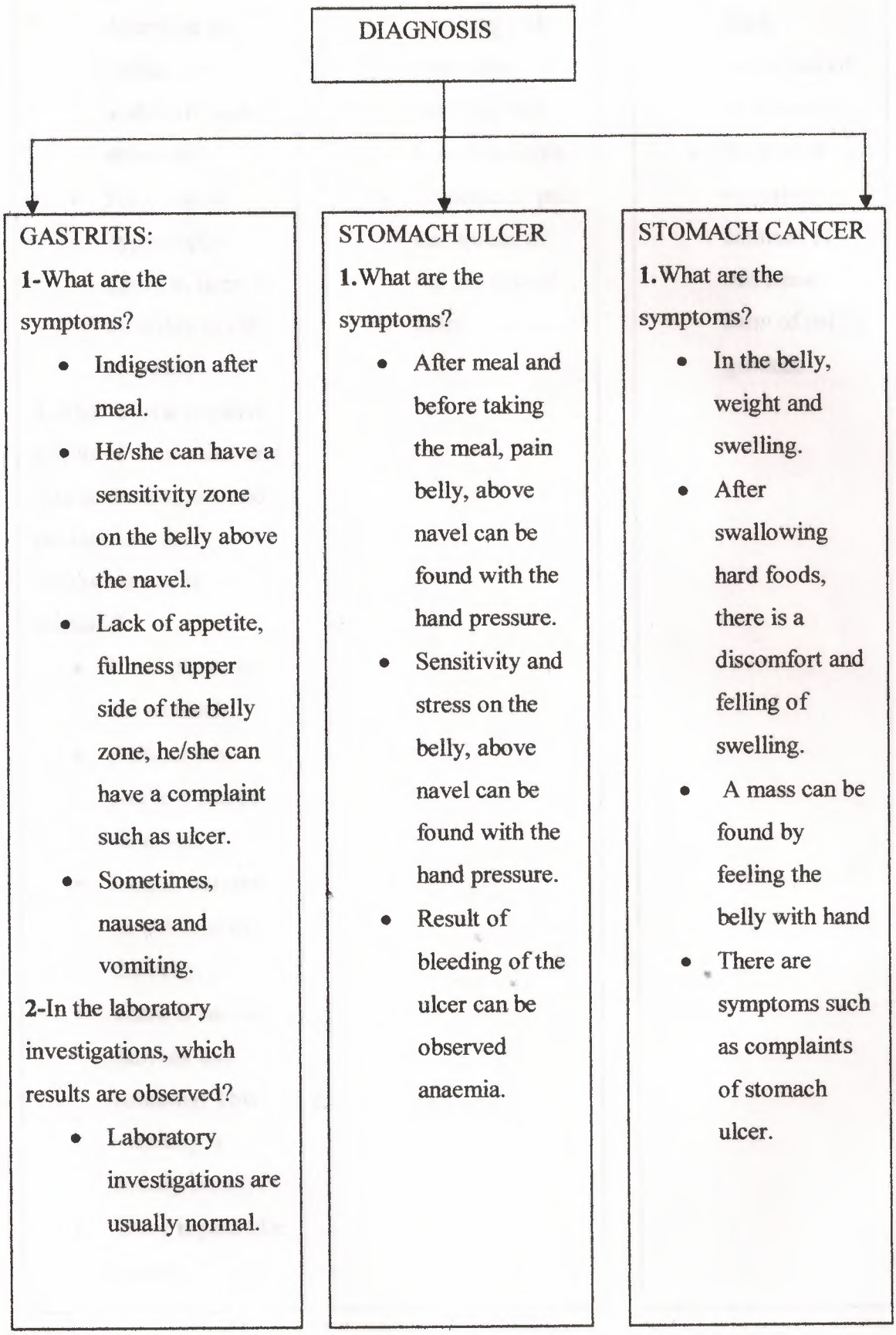


Figure 5.1. Simple diagram of expert system for medical diagnosis

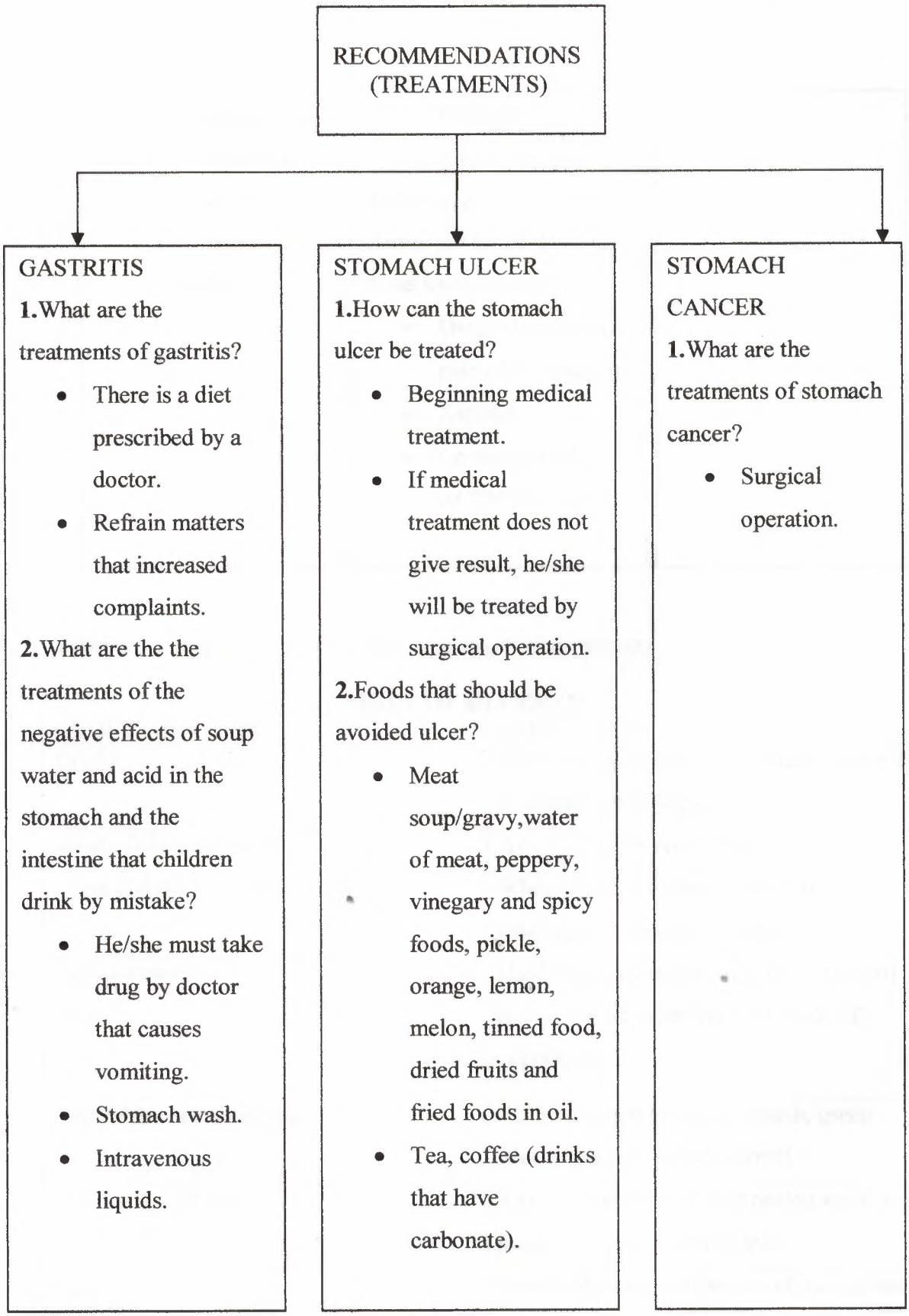
This expert system will behave as a doctor. And it checks inputs according to diagnosis (i.e. according to knowledge) and using this knowledge base, it will determine disease and recommendations for its treatment.

Table 5.1.Main diseases of the stomach and their diagnosis



• For atropric gastritis,they can determine the inadequacy of acid in the water stomach. • For choronic 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 5.2. Recommendations (treatments) of main diseases 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 5.3.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 5.3.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 5.3.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 5.4. 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 & VOMITING	Small (Small nausea)	Small	Large (At more intensity)	Very large
BLEEDING	No	Small	Middle (Anemia)	Large

Using the figure 5.2. we created table 5.1. 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”);

IF PAIN=SMALL
AND SENSITIVITY=SMALL
AND (NAUSEA_OR_VOMITING)=LARGE OR (NAUSEA_OR_VOMITING)=AT
MORE HIGH INTENSITY

AND BLEEDING=LARGE
THEN DISPLAY (“CANCER”);

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

IF PAIN=SMALL
AND SENSITIVITY=SMALL
AND (NAUSEA_OR_VOMITING)=VERY_LARGE
AND BLEEDING=MIDDLE_OR_BLEEDING=ANEMIA
THEN DISPLAY (“ULCER”);

IF PAIN=SMALL
AND SENSITIVITY=SMALL
AND (NAUSEA_OR_VOMITING)=VERY_LARGE
AND BLEEDING=LARGE
THEN DISPLAY (“CANCER”);

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

IF PAIN=SMALL
AND SENSITIVITY=MIDDLE
AND (NAUSEA_OR_VOMITING)=SMALL OR
(NAUSEA_OR_VOMITING)=SOMETIMES
AND BLEEDING=SMALL
THEN DISPLAY (“CHRONIC GASTRITIS”);

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

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

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

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

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

IF PAIN=LARGE
AND SENSITIVITY=VERY_LARGE
AND (NAUSEA_OR_VOMITING)=VERY_LARGE
AND BLEEDING=LARGE
THEN DISPLAY ("CANCER");

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

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

IF PAIN=VERY_LARGE
AND SENSITIVITY=VERY_LARGE
AND (NAUSEA_OR_VOMITING)=VERY_LARGE
AND BLEEDING=MIDDLE OR BLEEDING=ANEMIA
THEN DISPLAY ("CANCER")

CONCLUSION

In practice some of processes are characterized by hard formalized and unpredictable information, 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 expert system is considered within this project.

The architecture of an expert system for medical diagnostic 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 diseases is developed. This knowledge base contains nearly 20000 production rules. Premise parts of the rules include the main input characteristics of diseases, whereas the conclusion parts are the diagnosis and recommendation for treatment of illness. After defining diagnosis the system provide recommendation for the treatment of illness. The procedures for interpreting the knowledge base rules are developed.

The realization of the expert system is performed using expert system shell ESPLAN. The obtained results satisfy the efficiency of the applied methodology.

REFERENCES

1. AI WEEK, "Big Eight Accounting Firm Develops Tax ES", AI WEEK, July 1, 1988a, p.4.
2. Rafik Aliyev, Fuad Aliyev and Mamedgasan Babaev. Fuzzy Process Control and Knowledge Engineering in Petrochemical and Robotic Manufacturing, Verilag TUV Rheinland, 1991, p.146.
3. Aikens, J.S., J.C.Kung, E. H. Shortlife, and R. J. Fallat, "PUFF: An Expert system for Interpretation of Pulmonary Function Data, "in B. C. Clancey and E. H. Shortlife, (eds.), reading in Medical Artificial Intelligence: The First Decade, Addison-Wesley, Readings, Mass., 1984
4. Anderson, D., and C. Ortiz, "AALPS: A Knowledge-Based System for Aircraft Loading, "IEEE Expert, Winter 1987, pp. 71-79
5. Benchimol, G., P. Levine and J.C. Pomerol, Developing Expert Systems for Business, North Oxford Academic, London, 1987
6. Bonissone, P. P. and H. E. Johnson, "Expert System for Diesel Electric Locomotive Repair"
7. Brazile, R.P. and K.M.Swigger, "GATES: An Airline Gate Assignment and Tracking Expert System, "IEEE Expert, Summer 1988, pp. 33-39
8. Carter, C. and J.Catlett, "Assessing Credit Card Applications Using Machine Learning", IEEE Expert, vol.2, no 3, fall 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. Ignizio and A. L. Soyster, "Discriminant Analysis via Mathematical Programming: On Certain Problems and Their Causes", Computers and Operations Research, vol.16, no 4,1989,pp.71-79
11. Feigenbaum, E.A., Interview in Knowledge-Based Systems: A Step-by-Step Guide to Getting Started, Second Annual AI Satellite Symposium, Texas Instruments, 1986
12. Huntington, D., EXSYS Expert Systems Development Package, EXSYS Manual, Albuquerque, New Mexico, 1985
13. Ignizio, J. P., "Identification of Incompleteness in Knowledge Bases via a Rule Dependency Matrix", technical paper, Department of Industrial Engineering, University of Houston, July 1987

14. Ignizio, J. P., "Attribute-Value Pair Tables and Rule Base Architecture", technical paper, University of Houston, Houston, Tex., 1988
15. Lindsay, R.K.,B. G., Buchanan, E.A. Feingenbuam and J. Lederberg, Applications of Artificial Intelligence for Chemical Inference: The DENDRAL Project, McGraw-Hill, New York,1980
16. Pople, H.E. ,"CANDUCEUS: An Experimental Expert System for Medical Diagnosis", in P.H.Winston and K.A.Prendegast, The AI Business, M. I. 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 of the Recognition Process", UCAI-3, 1973,pp.185-193
18. Shortlife, E.H., Computer-Based Medical Consultations: MYCIN, Elsevier, New York, 1976
19. Ignizio, J.P.,Introduction To Expert Systems The Development And Implementation Rule-Based Expert Systems,Mc Graw Hill,1991 pp.127-147