The application of expert systems in parenteral nutrition

Michael Robinson
University of Wollongong

THE APPLICATION OF EXPERT SYSTEMS IN PARENTERAL NUTRITION

A thesis submitted in partial fulfilment of the

requirements for the award of the degree

Master of Science (Honours)

(Computing Science)

from

THE UNIVERSITY OF WOLLONGONG

by

Michael Robinson, BSc., Grad. Dip. Comp. Sci.

DEPARTMENT OF COMPUTER SCIENCE
1991.

010325

This is to certify that this thesis has not been submitted for a degree in any other University or Institution.

Michael Robinson

20th December, 1990.

ABSTRACT

Total Parenteral Nutrition (TPN) is a medical technique used to provide a patient's nutritional requirements via intravenous feeding. Critically ill patients must have adequate nutrition but must also have a stable physiology compensated for or treated by drugs.

Several factors such as the complex nature of the TPN solution, the cost of the ingredients and the possible interaction of nutrient and drugs has led to the development of small expert system to assist the hospital medical staff in formulating the TPN constituents and assist the pharmacy staff in producing the final solution.

This text will describe a small knowledge-based diagnostic system which when combined with conventional programming techniques has led to tangible benefits within a hospital Intensive Care Unit and Pharmacy.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

A number of people have contributed to the successful completion of this project.

For his guidance in research and preparation of this thesis, I would like to thank my supervisor Dr. N.A.B. Gray, for his constant support and encouragement throughout the course of this research.

Acknowledgements are also made to the Illawarra Area Health Service staff for their helpful assistance.

Finally, not the least acknowledgement goes to my wife, Rosalind, for her support and consideration throughout the course of my research.

CHAPTER 1

INTRODUCTION AND SUMMARY OF TOPIC

1.1 Parenteral Nutrition and Expert Systems

Total parenteral nutrition is defined as the provision of a person's nutritional requirements via any intravenous catheters, that is, a form of intravenous feeding. This technique had its origins with attempts to provide nutrition when the normal digestive system was inoperative for prolonged periods but the rest of the patient was relatively healthy. More recently, there is an awareness that unhealthy people require adequate nutrition and these form a second group of patients. By providing and manipulating the nutrients parenterally, when normal oral intake or gut function is impaired, the return to health is actively promoted. When the problem is solely the act of ingestion, external feeding via naso-gastric tube is used.

Critically ill patients, usually found in the intensive care unit (ICU) of a hospital, comprise the third group. In these patients not only is nutrition vital but manipulation of internal body chemistry is one of the central concepts of intensive care therapy.

Within the ICU, where frequent monitoring of various parameters is performed, immediate adjustment to patient response and alteration of parenteral therapy are an integral part of patient management.

Critically ill patients must have adequate nutrition but also have varying fluid and electrolyte physiology, which must be compensated for or treated with drugs. At any time, due to the results of laboratory tests or clinical monitoring, components of the solutions may have to be changed without altering the effect on other components within that solution.

The following major problems exist with the use of this technique in the ICU :-

1) Unless the pharmacy, where the solutions are made up, is manned 24-hours a day by a trained pharmacist, (this does not commonly happen in practice except in large metropolitan teaching hospitals) then alterations to any of the components of a solution in the light of clinical monitoring may have to be made by the ICU staff who are not fully trained in pharmaceutical practice or ICU staff must wait for the pharmacist on-call to arrive to prepare the solution.

2) The registrar (senior medical officer) has the legal authority to alter solutions but whilst authority to alter solutions may be delegated, in circumstances in which he is not always immediately available, his legal authority cannot be delegated.

3) Mixing techniques in the phramacy requires very strict adherence to written detail which is only learnt by practice. The order of addition together with amounts of ingredients must be closely controlled and taken into account when ordering. On average, it takes about four weeks for a resident medical officer (RMO) to become familiar with the reqirements and RMO's are replaced on a rotation basis every ten weeks.

4) TPN solutions are very expensive and have a shelf life when mixed of less than 36 hours.

5) The trained pharmacy staff expressed a need for additional computer-based productivity aids to complement the pharmacy dispensing and

stock control systems currently being implemented. Specifically, the main area of productivity gain was in the formulation of TPN solutions.

6) Some conventional programs had been developed to assist pharmacy staff, but these programs were found to be inadequate due to the difficulty in making changes to increasingly complex code and the overall inflexibility of the programs.

It was considered that the application of expert systems technologies was an ideal way to solve the above mentioned problems and perform functions similar to those normally performed by a human expert in that domain. For the TPN applications, it was determined that an expert system could support the data analysis aspect of the problem in two ways :

1) By providing a more intelligent interface and thus relieving the expert from the necessity of having in-depth knowledge about the mechanisms.

2) By capturing the expert's knowledge in a form that is storable and transportable, and thus provide the possibility of data analysis without the necessity of the expert carrying out the analysis in all cases.

It has been suggested [MCDONALD83] that data examination tasks within the medical environment individually consume minuscule amounts of time but collectively they take up much of the physician's day. It has also been suggested that to reduce the chance of error due to this "information overload" condition, a round-the-clock computer-based consultant would be of great assistance. The incorporation of expert system technology into diagnostic medicine applications has been successfully attempted in many projects since the late 1970s [RAUCH-HINDIN86]. The problems encountered in the production of TPN solutions in the pharmacy, together with the need to provide consultative medical assistance have led to the development of a consultative teaching expert system.

1.2 What is an Expert System?

During the past ten years, we have witnessed the emergence of a new computer technology called "expert systems". They allow a computer program to use expertise to assist in a variety of problems, such as diagnosing failures and designing new equipment. Utilising the results of artificial intelligence work on problem solving, expert systems have become a commercially successful demonstration of the power of artificial intelligence techniques. Correspondingly, by testing current artificial intelligence methods in applied contexts, expert systems provide important feedback to the science about the strengths and limitations of those methods.

The number of commercially available expert systems have grown from a few dozen five years ago, to a few thousand today. The CRI Directory of Expert Systems [CRI86] and Watermann [WATERMANN86] report on several hundred systems that are currently in use today.

Professor Edward Feigenbaum [FEIGENBAUM78] of Stanford University has defined an expert system as

"... an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution. Knowledge necessary to perform at such a level, plus the inference procedures used, can be thought of as a model of the expertise of the best practitioners of the field."

Expert systems are computer systems, comprising both hardware and software that mimic an expert's thought processes to solve complex problems in a given field or domain. An expert system manipulates information with the intention of solving a particular application problem. What makes expert systems unique is the way they approach and solve problems (combining knowledge representation and problem-solving algorithms) and the different types of problems they can solve (heuristic problems versus conventional problems). Expert systems attempt not only to apply conventional mathematical and boolean operators but to incorporate human reasoning processes such as "rules of thumb" to solving problems. The underlying goal of an expert system is to mimic an expert's thought processes in solving a problem.

Expert systems have the capacity to manipulate problem statements and integrate relevant pieces of knowledge from a knowledge base (a collection of information) using reasoning techniques, commonly known as heuristics, to emulate the expert.

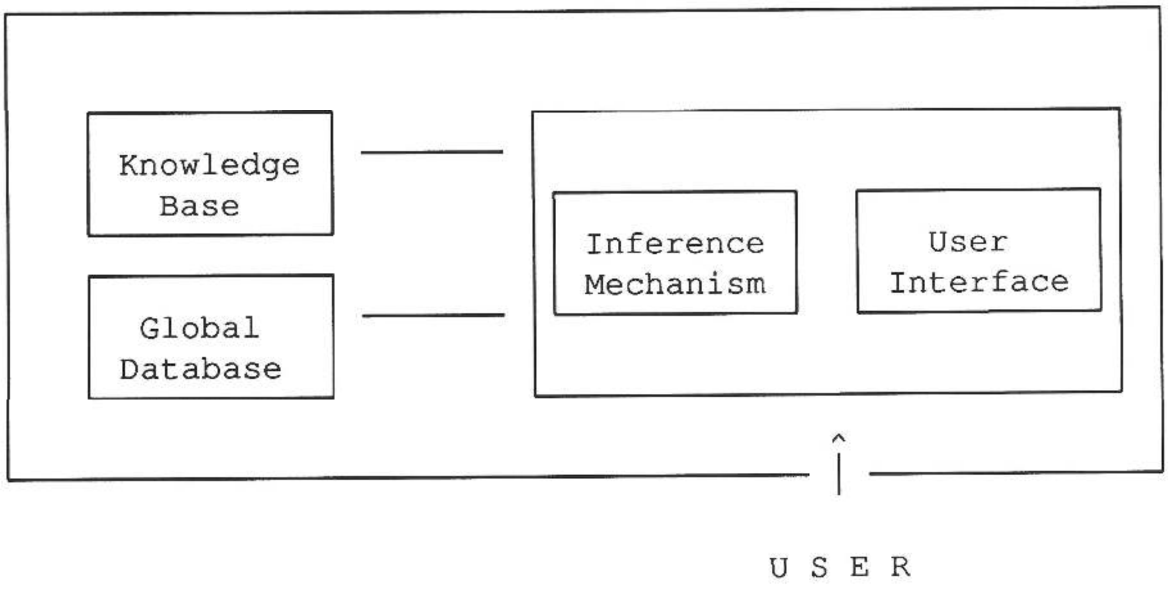


Figure 1 : Essential Components of an Expert System

The diagram shown in Figure 1 highlights the essential components of an expert system. These modules represent counterparts to essential factors of human expert reasoning :

1) The **Knowledge Base** corresponds to the knowledge and experience of an expert.

2) The **Global Database** contains existing data and hypotheses about the problem area.

3) The **Inference Engine** accommodates reasoning methods simulating the way human experts would apply their knowledge to analyse information and reach a decision.

4) The **User Interface** provides for communication between the user and the system.

The inference engine or mechanism performs the complex task of combining elements from the global database with elements from the knowledge base to produce new information in the form of advice. Facts gathered from information supplied by the user in response to questions asked by the system and information inferred by the system's inference mechanism working on the knowledge base are gathered in the specific data collection area, the global database. While collecting information supplied by the user into the global database is fairly straightforward, deducing new information from the old is subtle and complex. This responsibility belongs to the major component of an expert system, the inference mechanism.

1.3 Examples of Expert Systems

One of the first practical expert systems was DENDRAL [BUCHANAN84]. Work on DENDRAL began in 1965 at Stanford University, and its aim was to enumerate plausible structures in the form of atom-bond graphs for organic molecules from the information provided by analytical instruments and user-supplied constraints. The many possible structures to be explored required the use of exhaustive search in order to consider all possible solutions. The domain searched is not an explicit set of known organic structures, but an implicit set derived from a theory of how such structures are formed.

There have since been many other expert systems springing from DENDRAL. In the drive to create expert systems to model human thought processes, the medical consultant MYCIN [SHORTLIFFE76] and the geological expert PROSPECTOR

[DUDA80] were specifically designed to represent and explain all the reasonings in human terms, in the sense of presenting the sequence of rules used to achieve a conclusion.

In both cases, the principle of generating vast numbers of solutions from some theoretical model is inappropriate. Also, it is assumed that no physician would or should consider using a proposed therapy, nor would a geologist recommend expensive mining operations unless each could justify the conclusions of the relevant expert system.

All artificial intelligence programs, including expert systems, represent and use knowledge. The conceptual paradigm of problem solving that underlies all of artificial intelligence is one of search (ie., searching among alternative solutions). Although this concept is immediately clear and simple, it does not suggest how to search a solution space efficiently and accurately. For large solution spaces, such as the two examples described below, the number of possible solutions may be astronomical.

System	Description of Search Space	Solutions
MYCIN :	combination of between 1 and 6 organisms from a list of 120 organisms.	$> 6 * 10^6$

XCON :	arbitrary number of computer system components selected from more than 20,000 catalogue items at between 50 to 150 items at a time.	$> 10^{200}$

Most expert systems, however, use heuristics to avoid the exhaustive search, much the same as human experts. For problem areas in which experts are acknowledged to be more efficient and accurate than non-specialists, it is reasonable to assume that what the experts know can be retained for use by a computer program. This concept is one of the fundamental assumptions of knowledge engineering, the art of building expert systems by eliciting knowledge from experts [HAYES-ROTH83].

1.3.1 MYCIN - An Infection-Diagnosing System

MYCIN was one of the earliest expert systems in the medical field. The success it achieved in handling such difficult problems made it one of the classic expert systems, and its approach to diagnostic problems has become one of the standard paradigms for expert system consultation shells.

The program was developed strictly as a research system at Stanford University during 1970's to aid physicians in the diagnosis and treatment of meningitis (infections that involve inflammation of the membranes that envelop the brain and spinal cord) and bacteremia infections (involving bacteria in the blood).

The goal of medical diagnosis is to decide, based on the patient's symptoms and other data such as microbiological cultures and the patient's medical history, whether the patient has an infectious disease, and if so, what micro-organism might be causing it. The treatment problem is formulated in terms of what antibiotics to prescribe.

When the program is invoked, it initiates a dialogue with the physician and reasons about data associated with a patient, laboratory results, symptoms and general characteristics. Eventually, MYCIN provides a diagnosis and a detailed drug therapy recommendation.

What makes the type of problem MYCIN is intended to solve more difficult than it might at first appear is that patients often develop infections and infectious diseases after surgery, while they are still in a weakened state. In such a situation, the infection must be quickly eliminated or the patient may die. Physicians will usually consult a specialist under such circumstances to decide what to do in time in order to save the patient. MYCIN attempts to capture the knowledge of such a specialist.

The strategy used by MYCIN to solve diagnosis and therapy problems is a modified top-down, or backward-chaining approach. This means that MYCIN starts from the goal or hypothesis that is to be achieved; that is, to choose a prescription of antibiotics. First, it forms a hypothesis of a possible therapy and then proceeds to reason backward to the conditions that would have been true for this to be the correct prescription. There are many possible solutions to the problem and MYCIN rules are used to search back for the supporting facts.

1.3.2 XCON - A Planning and Design System

XCON, named R1 [1] at Carnegie-Mellon University (CMU) where it was originally developed, configures Digital Equipment Corporation (DEC) VAX-11 and PDP-11 computers. Given a customer's order, XCON determines what, if any, substitutions and additions need to be made to build a complete and functional computer system. In response to user input, XCON produces print-outs showing the components chosen, the reasons for adding or deleting others, cable types and lengths, device addresses and a series of diagrams showing the special and logical relationships among the modules and devices in a computer system.

1'Four years ago I couldn't even say "knowledge engineer", now I ...' [MCDERMOTT82]

The designer of XCON [MCDERMOTT82] suggested three main factors that contribute to making the computer configuration task difficult and are conducive to the development of an expert system to perform such a complex task :-

1) Determining completeness of the configuration. Some configurations are extensive and sometimes extremely complex.

2) Equally good configurers tend to configure the same set of components in different ways, and so the configurations tend not to be consistent.

3) It is difficult to keep skilled, experienced configurers, since they tend to be promoted. The skills however are an asset and need to be retained.

To perform its configuration tasks, XCON, like a human configurer, requires two kinds of knowledge; component knowledge and configuration constraint knowledge. Component knowledge is knowledge about each of the components that a customer might order, including information relevant to the computer configuration, such as voltage, frequency and how many devices the computer can support. Configuration constraint knowledge, in the form of IF-THEN rules, indicate what classes of components can or must be associated to form the system configuration. XCON uses its restraint knowledge to limit the number of combinations of components in order to form a functionally acceptable computer system.

XCON uses a different style of reasoning to that used by MYCIN, working forward from facts to other facts which they imply. This style of reasoning is called "forward-chaining".

The kind of reasoning shown in the MYCIN example is appropriate when you have a definite conclusion in mind and want to discover whether it is true. They are often called "goal-directed", because they have in mind a particular goal, to prove a certain fact.

In the XCON application however, the possible conclusions cannot be stated beforehand. In this case, a different, "data-directed" reasoning method is appropriate. The reasoning sequence stops not when a particular fact has been proven, but when as many facts as possible have been proven, i.e., the current state of data is matched against a rule or knowledge-base and instructions are followed and acted upon until all the current data has been checked.

The XCON rule-based constraint model described above is well suited to be used as a methodology for an expert system designed to perform TPN calculations. The TPN calculation cannot be subjected to a backward chaining methodology because there are too many ways of combining the ingredients for each combination to be specifiable as a conclusion to be tried.

The TPN calculation problem conforms to some of the criteria for a good OPS5 domain. There a small number of sets of symbolic data (patients and TPN solutions) that are fit together according to a set of constraints (medical and pharmacological guidelines). Using a brute-force method to enumerate all possible combinations of TPN solutions for each patient is infeasible. The heuristic approach, however, produces a satisfactory solution by describing the conditions under which the TPN solutions may be formulated.

1.4 Objectives and Structure of Study

The main objective of this study was to show the possibility of developing a consultative or teaching application using expert system technology. Building the expert system from scratch was considered to be a research topic by itself and so a high-level AI language OPS5 was used to develop the system. The second objective, therefore, became the testing of the applicability of OPS5 language to build the application. The third objective was to demonstrate the capability of an integrated TPN expert system against the human expert.

Basically, the following six-step plan was used in the development of the expert system :-

1) Select a development tool and implicitly make a commitment to a particular consultation paradigm or problem solving scenario.

2) Identify the problem and analyse the knowledge to be included in the system.

3) Design the system on paper.

4) Develop a prototype or prototypes and test these by running consultations.

5) Expand the prototype and revise until complete.

6) Maintain and update system as required.

Due to the complexity of the TPN ordering and manufacturing process, it was decided to separate the study into two sections :

1) Create TPN order based on patient details and consultation with the physician ; and

2) Calculate the final TPN solution ingredients based on existing pharmacological procedures and consultation with the pharmacist.

Preliminary research was carried out on the theory of expert systems, which is documented in Chapter 2. Based on this research, an expert system building tool, OPS5, was selected from those available. A brief description of the language is given in Chapter 3.

Chapter 4 describes the system itself, as it has been implemented during the study. After describing the implementation environment, the main emphasis is placed on the presentation of the interaction between the expert system and the conventional programming exterior. Chapter 5 presents an annotated version of selected parts of the system.

Chapter 6 evaluates the performance of the expert system. The concluding Chapter 7 summarises the research work done for the project. It describes the advantages and disadvantages of the method chosen for the implementation. The report ends with recommendations for future work on expert system integration and enhancements for other areas of patient management.

CHAPTER 2

ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS

2.1 Introduction

In the fledgling field of computer science known as artificial intelligence (AI) during the late nineteen sixties, scientists decided there must be another way to make a computer program intelligent. If it was too difficult to make the entire program general purpose, then they would concentrate instead on developing techniques to use on specialised programs.

During this period, the AI scientists concentrated on representation (how to formulate the problem) and search (how to find a solution). In the early nineteen seventies, it was realised that the problem-solving power of a program comes from the knowledge it possesses. This realisation led to the development of special-purpose computer programs, systems that were expert in some narrow problem area. These programs are called expert systems, or for less specific applications, knowledge-based systems.

The field of AI involves the creation of computer systems which exhibit behaviour that, when observed within humans, can be deemed to show intelligence. AI researchers attempt to uncover the underlying theory of intelligence, to create a "science of intelligence".

But what is meant by the term intelligence? Is it the ability to solve complex problems quickly or to solve problems with a high degree of accuracy?

While both of these are characteristics attributed to beings that are viewed as intelligent, are they sufficient? While many humans have an adequate subjective concept of what is meant by this term, few can provide a precise definition. Webster's Dictionary [WEBSTER76] defines intelligence to be :-

"the available ability... to use one's existing knowledge to meet new situations and to solve new problems, to learn, to foresee problems, to use symbols and relationships, to create new relationships, to think abstractly."

This definition presents key elements of intelligent behaviour : the ability to represent knowledge and the ability to use that knowledge in problem solving.

2.2 Principles and Applications of AI

In spite of the fact that computers were originally built as numerical processors, a small group of computer scientists continued to explore the ability of computers to manipulate non-numerical symbols. Simultaneously, psychologists concerned with human problem solving sought to develop computer systems that would simulate human behaviour. Over the years, individuals concerned with symbolic processing and human problem solving have formed that inter-disciplinary subfield of computer science called artificial intelligence (AI). AI researchers are concerned with developing computer systems that produce results that we would normally associate with human intelligence.

Chess playing was one of the first AI applications that was successfully implemented. It displays one of the fundamental principles of AI, the concept of intelligent search. For each move in a chess game, there are many choices. This leads to a combinatorial explosion of possibilities through an average game of approximately 40 moves (approximately 10^{120} possibilities) [KUMARA86]. Testing all possible alternatives cannot be achieved even with today's supercomputer hardware and the search must be restricted to be practical. A search-limiting algorithm, such as 'alpha-beta pruning' [MCCARTHY83], stops examining possible consequences of a chess move as soon as it finds one situation refuting the move and then heuristic rules are employed to reject most alternatives.

The most widely studied issues in AI research include :

1) what information a program should have and how to store the information in the computer; and

2) how further conclusions can be drawn from the initial information.

Mathematical logic provides many powerful methods for both the representation of knowledge and ways of reasoning. Logic studies the relationship of implication between assumptions and conclusions and gives formalisation to a systematic way of reasoning. Additionally, mathematical models representing real-world reasoning using fuzzy logic make it possible to simulate the way humans are perceived to reach conclusions when confronted with incomplete or imprecise initial information.

The field of artificial intelligence can be subdivided into several sub-areas :

1) **Pattern Matching**

This includes systems that are capable of recognising objects by comparison with stored patterns kept within a database. The system systems should be able to identify identical and similar objects. Changes in lines, colours, intensity are used to categorise specific features of objects and correlate the resulting feature vector to the feature space stored in the database. In the field of surveying, pattern recognition is of interest to identify objects in remotely sensed scanned data or scanned maps.

2) **Natural Language Processing**

This includes systems using a grammar syntax and a dictionary with a semantics interpreter. The validity of sentences is checked against predefined grammar rules, then the semantics interpreter analyses the meaning of a sentence. Practical applications include comprehending text, language translation and an intelligent interface for database queries.

3) **Robotics**

This field of research includes machines designed to carry out strenuous or dangerous tasks which cannot be performed or it is not desirable to be performed by humans. Applications include factory material handling, combat and combat support, planetary exploration and industrial processing such as welding and painting. Today's robots deal with very specialised tasks, but will in the future be easily programmable, general purpose systems.

4) **Expert Systems**

This sub-area includes computer programs applied to emulate reasoning processes requiring expert knowledge and experience. The systems consist of a database of data and knowledge and a system that controls the application of this knowledge to analyse the data. Expert systems have been successfully employed in areas of expertise such as medical diagnosis, mineral exploration and computer configuration. Interaction between expert systems and existing large databases is currently a very active area of research.

Interaction between these topics are common since any automated system attempting to simulate human performance must be able to act intelligently in more than one narrowly defined area. For example, most robotic systems use pattern matching as a means of analysing visual data.

2.3 Expert System Technology

The most successful application of AI is knowledge-based or expert systems. Expert systems are knowledge-intensive computer programs. They contain knowledge about their speciality and use rules of thumb, or heuristics, to focus on the key aspects of particular problems and to manipulate symbolic descriptions in order to reason about the knowledge they are given. The systems often consider a number of competing hypotheses simultaneously, and they frequently make tentative recommendations or assign levels of confidence to alternatives.

The term "expert system", however, tends to be misleading. The connotations of the word "expert" can lead to expectations of infallible performance which we do not insist upon in those we call experts. It follows that we should not demand more of our expert systems than we ask of human experts (although some expert systems have in fact been seen to outperform a human expert in some areas).

It has been recognised that the power of the expert system does not necessarily rest in its reasoning power or its architecture, but rather in the structured knowledge that it contains. Hence, the term "knowledge-based systems" (KBS) has become an apt description for many expert systems projects. In fact, since there is today a bias among some expert systems developers that true "experts" can rarely, if ever, be cloned in machine form, the new KBS terminology is normally favoured over the more popular term of "expert system". The KBS terminology not only reduces the sometimes unrealistic expectations suggested by the word "expert", but also points to the differences between such systems and conventional programs. In this text, however, the terms will be used interchangeably.

A rule-based system is a knowledge-based system in which some variation of the production rule [DAVIS77] is used to encode knowledge. The term "expert system" has been used to refer to knowledge-based systems which perform tasks usually performed by "experts". The vast majority of these expert systems to date have in fact been rule-based systems [WATERMANN86].

Enthusiasts see expert systems as a replacement for scarce human experts, the embodiment of human wisdom, a repository for the collective knowledge of many experts, or tools to enable novices to behave or perform like experts. Many critics of expert systems complain that the promises are far more attractive than the resulting reality and that the term "expert systems" is seductive and possibly misleading.

During the evolution of expert systems, it has become apparent that data search strategies alone, even when augmented with detailed heuristic evaluations, are often inadequate for solving real world problems. The complexity of those problems was usually such that, without incorporating substantially more problem knowledge than had been acquired, the realistic search space size could not be generated. It quickly became apparent that for many problems, expert domain knowledge was even more important than the search strategy or inference procedure. This realisation has led to the field of knowledge engineering, which focuses on ways to bring expert knowledge to bear in problem solving.

One aspect of the knowledge-based approach is that the complexity associated with real-world problems is reduced by the more powerful focussing of the search that can be obtained with the more commonly used rule-based heuristics. The rule-based system is able to reason about its own search effort, in addition to reasoning about the problem domain.

Today's expert systems normally have the following three characteristics :

1) They deal with a specific, focussed task having a relatively narrow range of applicability.

2) Knowledge is kept separate from the reasoning methods used to draw conclusions.

3) They are able to explain their actions and lines of reasoning, if required.

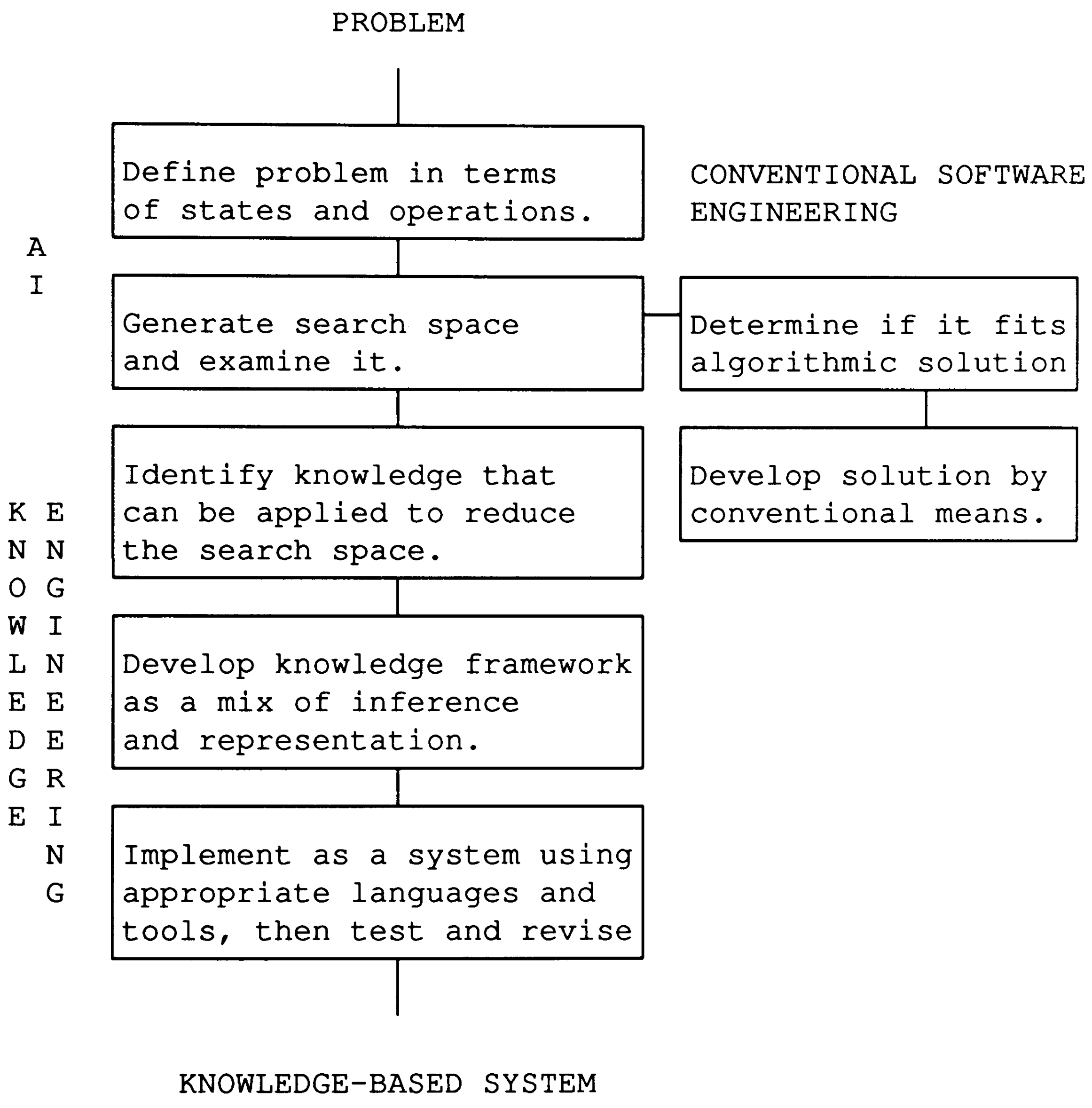

PROBLEM
Define problem in terms of states and operations.
CONVENTIONAL SOFTWARE ENGINEERING
A I
Generate search space and examine it.
Determine if it fits algorithmic solution
Identify knowledge that can be applied to reduce the search space.
Develop solution by conventional means.
KNOWLEDGE ENGINEERING
Develop knowledge framework as a mix of inference and representation.
Implement as a system using appropriate languages and tools, then test and revise
KNOWLEDGE-BASED SYSTEM

Figure 2 : Interrelationship between different techniques

AI is a research field concerned primarily with studying problem solving in the abstract. Knowledge engineers, on the other hand, focus on replicating the behaviour of a specific expert when he or she is engaged in solving a narrowly defined problem. The interrelationship between AI, knowledge engineering and conventional programming is shown in Figure 2.

Many regard this shift from the study of generic problem-solving techniques to a focus on building systems that contain large amounts of specific knowledge about a particular problem as a major conceptual breakthrough in AI in the last fifteen years.

2.3.1 Expert System Concepts and Techniques

The diagram below shows the different modules which make up an expert system. These modules represent counterparts to essential factors of human reasoning.

The knowledge base corresponds to the knowledge and experience of an expert, whereas the global database contains existing data and hypotheses on the problem area. The inference mechanism accommodates reasoning methods simulating the way human experts would apply their knowledge to analyse information and reach a decision. The user interface provides for a communication path between the user and the system.

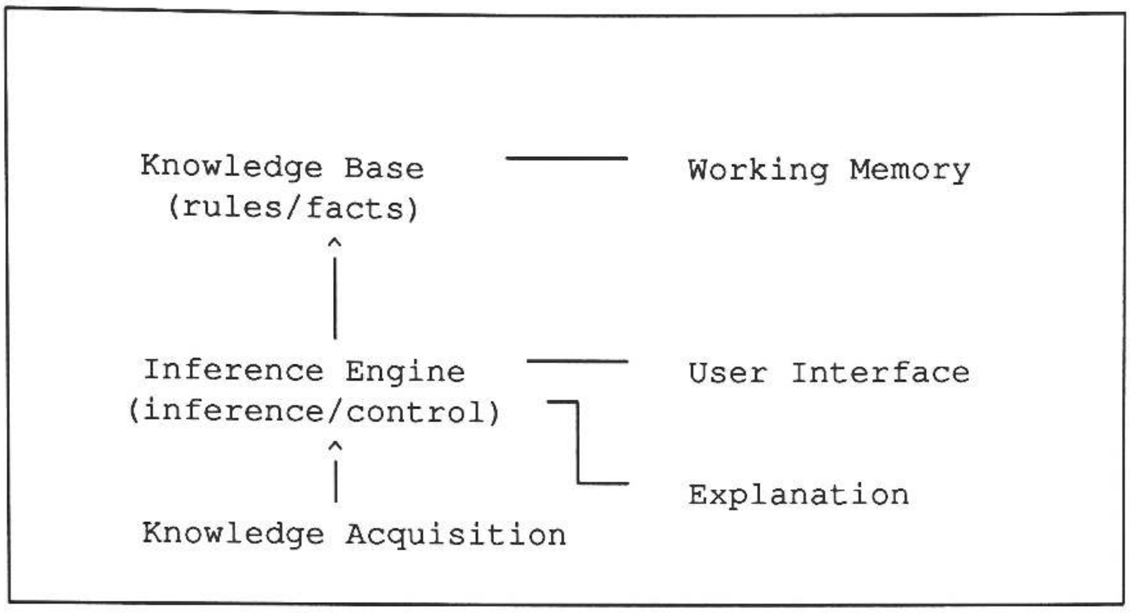


Figure 3 : Basic Modules in an Expert System

The inference mechanism performs the complex task of combining elements from the data base with elements from the knowledge base to produce new information in the form of advice. Facts gathered from information supplied by the user in response to questions asked by the system and information inferred by the system's inference mechanism working on the knowledge base are gathered in a specific data collection area, the global data base. It is this separation between the knowledge base and the inference mechanism which makes expert systems more versatile than conventional computer programs. Knowledge and control of the system can be built and maintained separately.

The global database provides a working storage during the evaluation of rules. At the beginning of an expert system session, the database usually contains a hypothesis to be proven and some facts known initially about the problem. The information is constantly updated and stored in the database temporarily until the end of the run.

In programs that combine knowledge from different expert systems, the global database is referred to as the blackboard. The information gained or derived from different programs is stored and retrieved from here, with the blackboard acting as the link of data flow between the systems.

2.3.2 Knowledge Representation

There are several standard ways that knowledge can be structured in a program, any of which can be used alone or in conjunction with others to build expert systems. Each technique has certain benefits, such as making it more easily understood, more easily modified or more efficient. The most widely used techniques are :-

1) Semantic Networks

2) Frames

3) Rules

4) Logic

2.3.2.1 Semantic Networks

In contrast to conventional database mechanisms that represent relatively straight forward relationships, knowledge bases contain a system of symbolic data structures which have meaning or semantics built in. The semantic network or semantic net is one of the oldest and most general representation schemes in AI. A semantic network is a graphical notation that represents objects (or actions or events) and contains built-in real-world meaning about the objects. The arcs or links between the nodes represent attributes that indicate relationships between the objects shown in the nodes.

Nodes are used to represent objects and descriptors. Objects may be physical, conceptional or abstract entities. Descriptors provide additional information about the objects. Links relate the objects and descriptors in the form of is-a to represent a class (instance of) relationship or has-a to identify nodes that are properties of other nodes. Further links are definitional and others may capture heuristic knowledge.

Those expert systems which can store knowledge as semantic networks also have a special part of the inference engine devoted to operations like inheritance. One of the first expert systems to use semantic networks was PROSPECTOR [DUDA80]. Part of a PROSPECTOR semantic network for mineral classification is shown in Figure 4.

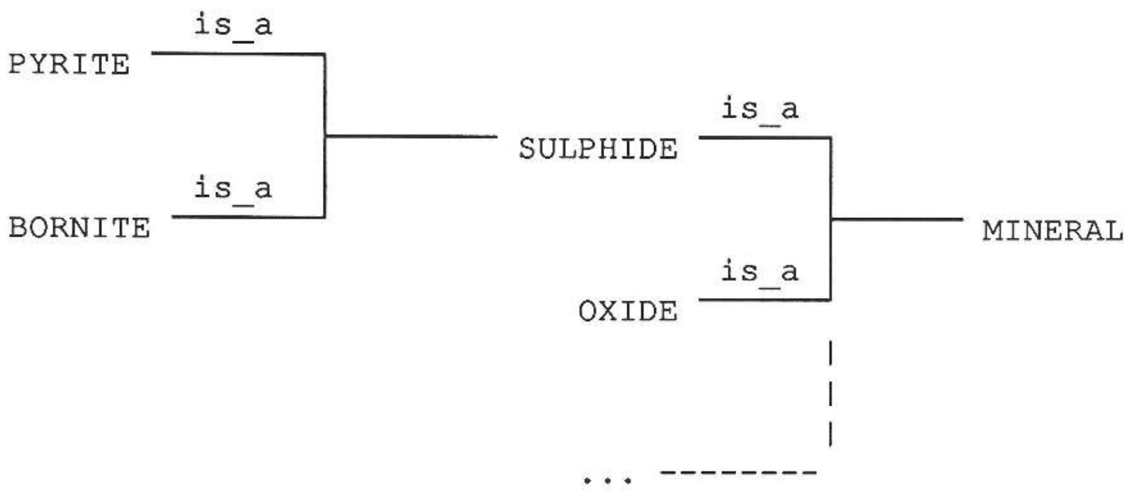


Figure 4 : Example of a part of a PROSPECTOR semantic network for mineral classification.

PROSPECTOR also has rules like :-

IF pyrite in veinlets is present THEN ...
IF sulphides are present THEN ...

If the user replies 'yes' to the first conditional statement, then the second conditional is automatically true, because from the information contained in the PROSPECTOR semantic network shown in Figure 4, the inferencing mechanism can deduce that pyrite is in fact a sulphide.

Factual information may also be represented as object-attribute-value (O-A-V) triplets. This scheme is used in MYCIN and is a specialised case of the semantic network approach. The exotic links that may exist between nodes in the semantic network are replaced with simple relationships. The object-attribute link is a "has-a" link and the attribute-value link is a "is-a" link.

The major advantage of using semantic networks is that they give a good structural overview of the relationships involved. They have, however, been criticised over the lack of meaning expressed by the arcs and nodes.

2.3.2.2 Frames

Another symbolic knowledge representation structure used to represent knowledge in knowledge bases is known as frames (or schemas). It is generally considered that both frame and semantic nets are frame-based representation methods used to represent facts and relationships. Frame-based knowledge representation uses a network of nodes connected by relations and organised into a hierarchy. Each node in the network represents a concept that may be described by values and attributes that are associated with that node. Nodes may inherit properties from other nodes according their hierarchy within the network.

In the same way that conventional databases store information in memory areas called fields, the frame uses a symbolic representation of an object that contains memory areas called slots for all the information associated with the object. Slots, like attributes, may store values, but may also contain default values, links to other frames, sets of rules or procedures. The inclusion of these additional features allow frames to offer a more flexible representation of knowledge at the expense of being more complex and more difficult to develop than the simpler O-A-V or rule-based systems. The relationships that can be indicated with hierarchies of frames are similar to those shown in semantic networks, but they are different in the way programmers view the knowledge - groups of text and pointers instead of graphical nodes and arcs.

2.3.2.3 Rules

Although there are many ways to represent knowledge, most expert systems store their knowledge in the form of rules and are commonly referred to as rule-based systems. Rule-based knowledge representation centres around the use of conditional statements that specify an action that takes place under a certain set of conditions. In some respects, rules are similar to conditional statements in the form of:-

IF (condition...) THEN (action...)

statements. The left-hand side of the rule is comprised of one or more conditions and the right-hand side consists of one or more propositions or consequents.

The matching of rule IF portions to the facts can produce inference chains. These inference chains indicate how the system used the rules to infer its conclusions.

Rules provide a natural way for describing processes driven by a complex and changing environment. A set of rules can specify how the program reacts to the changing data without requiring detailed advance knowledge about the flow of control.

The difference between rules and IF-THEN statements is major and has significant implications for the building of expert systems. In fact, despite their similarities, experience has shown that it is very difficult - often impossible - to develop a knowledge-based system using conventional IF-THEN statements. The difficulties exist because knowledge systems require greater conditionality and greater need to make changes during the program development than do conventional software systems. Writing programs that satisfy the greater conditionality and flexibility requirements of knowledge-based systems is hampered by using conventional branching techniques.

In a conventional program, the flow of control and use of data are pre-determined by the program's code and processing takes place in sequential steps, and branches when required. Where the problem is data-driven however, where branching is the norm, not the exception, rules offer the opportunity to examine the current state of the world at each program step and react appropriately.

A number of deficiencies have arisen with the use of rule-based systems [WILLIAMS89], which include restrictions imposed by the knowledge representation scheme, the encoding of much implicit knowledge, the knowledge acquisition bottleneck [WATERMANN86], and the inability to add in new types of knowledge to the system after the system has been developed.

More sophisticated mechanisms for representing knowledge for use by an expert system have thus been sought to store the general knowledge, rules and uncertain information that differentiates knowledge-bases from databases.

2.3.2.4 Logic

Logic is the most well defined way of expressing and reasoning about knowledge. In this section, a brief introduction to predicate logic will be presented.

Propositions are used in logic to express knowledge. A proposition can be either true or false and may be combined using connectives. Given two propositions, P and Q, the following relationships can be formed :-

conjunction :	P^Q to express that both P and Q are true.
disjunction :	PvQ to express that either P or Q is true.
implication :	P=>Q to express that Q is true when P is true.
equivalence :	P<=>Q to express that P is equivalent to Q.
negation :	NOT P to express that P is false.

The above rules can be used recursively to form more complex statements. For example,

The sun is round ^ The sun is red-hot

The sun is red-hot => There is no life on the sun

Via the ^ elimination inference rule followed by the => elimination inference rule provided by logic, we can infer that :-

There is no life on the sun

This language of propositions together with the rules of inference is called propositional calculus. The language has the property that is decidable. That is, there is an algorithm which given any complex statement conforming to the syntax rules of logic will decide whether the statement is true or false.

2.3.3 Knowledge Acquisition

The process of extracting knowledge from an expert and structuring the knowledge into rules or frames is called knowledge acquisition. The type of knowledge used by experts to solve problems is often subjective, partly judgmental and ill-defined. In most cases, it is not formulated in a fashion that is easily translatable into a computer program. The difficult task of extracting the expert's understanding of a problem and representing it as facts and relations in an expert system is often performed by a knowledge engineer.

This transfer of knowledge from an expert to the computer program may be represented by the following diagram :-

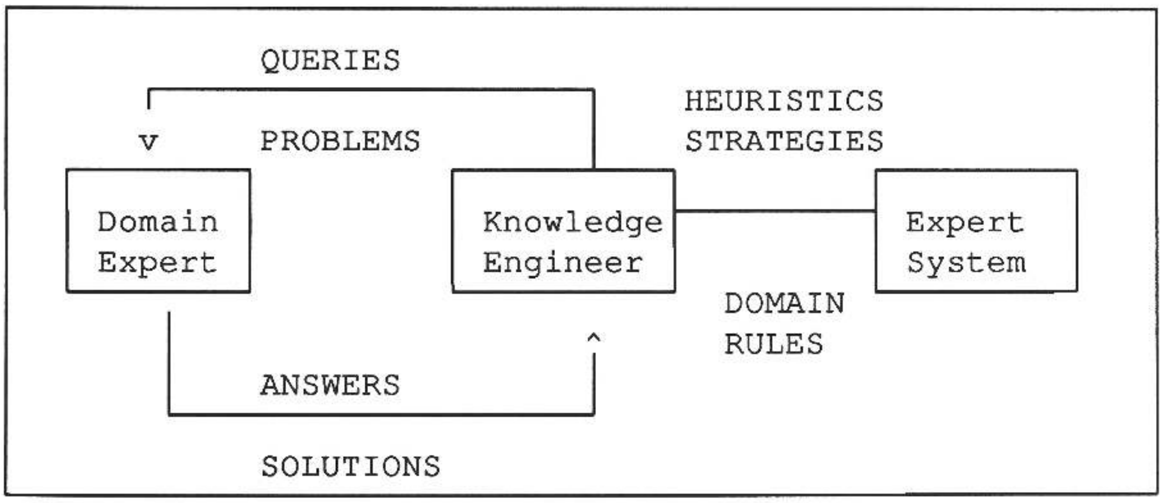


Figure 5 : Knowledge Transfer

Knowledge engineering relies heavily on the study of human experts in order to develop intelligent, skilled programs. As Hayes-Roth and others [HAYES-ROTH83] explain :-

"The central notion of intelligent problem-solving is that system must construct its solution selectively and efficiently from a space of alternatives. When resource-limited, the expert needs to search this space selectively, with as little unfruitful activity as possible. An expert's knowledge helps spot useful data early, suggests promising ways to exploit them, and helps avoid low-payoff efforts by pruning blind alleys as early as possible. An expert system achieves high performance by using knowledge to make the best use of its time."

The major barrier found in designing the system is the task of assembling and codifying the knowledge. Feigenbaum [FEIGENBAUM78] states that :-

"There are many important problems of knowledge representation, utilisation and acquisition that must be solved, but the acquisition problem is the most critical 'bottleneck' problem.".

Watermann [WATERMANN86] describes that the acquisition bottleneck difficulty arises because an expert

"has the tendency to state (his) conclusions and the reasoning behind them in general terms that are too broad for effective machine analysis...the pieces of basic knowledge are assumed and combined so quickly that it is difficult for him to describe the process."

2.3.4 Drawing Inferences

While collecting information supplied by the user into the global data base is fairly straightforward, deducing new information from the old is subtle and complex. This responsibility belongs to the major component of an expert system, the inference mechanism. Inference and control strategies guide a knowledge system as it uses facts and rules stored in its knowledge base, and the information it acquires from the user.

The most common inference strategy used in knowledge systems is the applications of a logical rule called modus ponens. This rule says, as we all do without thinking about it, that when A is known to be true and if a rule states "if A, then B", then it is valid to conclude that B is true. Stated differently, if the premise of a rule is true, then we are entitled to believe the conclusions.

2.3.4.1 Reasoning about Uncertainty

Information used to derive a decision which solves a problem is often afflicted with uncertainty. In addition, an expert often has to solve a problem without having a sufficient amount of data, using conflicting information or unreliable knowledge for interpreting the data. In order to simulate the human decision making process, an expert system must have some "feel" for the certainty of its knowledge and must be able to cope with these uncertainties on its path of reasoning.

There are many approaches for representing uncertain knowledge, none of which are universally applicable. Three of the most common methods are based on :-

1) Probability-based

2) Fuzzy Logic

3) Qualitative.

The most straightforward approach (as used in MYCIN) and the most popular, provides a subjective probability for each proposition. Using 'Bayesian inference', the probability for a particular event, given some set of observations or conditions, can be calculated. Using the notation P(X|Y) to mean the subjective probability ox X occurring given that Y has occurred, Bayes' rule states :-

$$P(X|Y) = \frac{P(X) * P(Y|X)}{P(Y)}$$

Approaches which use Bayes' rule as a basis of combining evidence have some drawbacks. Firstly, there is no indication of whether the probability was a wild guess or a judgment based on experience. The single value tells us nothing about its precision.

Secondly, a single value combines the evidence for and against a proposition without indicating how much there is of each. Despite these criticisms, several simple forms of 'Bayesian inference' are widely used by many existing expert systems.

The rules in the knowledge base frequently contain a "pseudo-probabilistic" indication of their degree of certainty, called certainty or confidence factors (CF), usually on a scale of -1 to +1, 1 to 10 or sometimes 0 to 100. Consider the following two rules and their associated certainty factors :-

IF winter THEN high temperature on a given day is less than zero degrees centigrade (CF=0.8) and

IF high temperature on a given day is less than zero degrees centigrade THEN it will snow (CF=0.3)

That is, suppose that on a winter day we are 80 percent confident that the temperature will remain below freezing, and we are 30 percent confident that on a day when the temperature remains below freezing, it will snow. The question that follows would naturally be : "How confident are we that it will snow on a given day in winter?"

One solution is to multiply the certainty factors of the two rules involved. This would result in the new rule :-

IF it is a winter day THEN it will snow (CF=0.24)

Generally, the system for propagating uncertainty can be summarised by :-

If a rule is of the form IF A THEN B (CF=X) and another rule is of the form IF B THEN C (CF=Y), then these rules may be combined to generate a third rule of the form IF A THEN C (CF=XY).

A second solution for uncertainty propagation works in the following way :-

If a rule is in the form IF A THEN B (CF=X) and another rule is in the form IF C THEN B (CF=Y), then these rules combine to generate a third rule of the form IF A AND C THEN B (CF=X+Y-XY).

A third technique combines the certainty factor associated with a rule together with the confidence expressed by the user in order to determine an overall level of confidence :-

If a rule is of the form IF A THEN B (CF=X) and if the user's confidence in fact A is Y, then the confidence in fact B is XY.

A second approach, used by Prospector [DUDA80], uses two separate values for the validity of each proposition, a measure of belief and an independent measure of disbelief. The two values are combined to give a single assessment of the proposition termed a 'certainty factor'. This method is preferred to the first because it overcomes the for and against evidence seen in the previous method.

Finally, another approach is fuzzy logic. Fuzzy logic is a technique which provides a way of representing fuzzy or continuous properties of objects. To accommodate informal arguments, fuzzy set theory provides a framework in which membership of a category is graded rather than simply definite. An example of a fuzzy proposition, where x is a large number, for which the fuzzy set may be constructed as :

	Likelihood
x between 0 and 10	0.1
x between 10 and 1000	0.2
x greater than 1000	0.7

Fuzzy logic takes this a step further by arguing that there is no clear boundary between the statement being true and a statement like 'unlikely', 'possible' and 'likely' as well as 'impossible' and 'certain'. This follows the realisation that human experts do not think in terms of numerical certainty factors. A doctor is more likely to say, 'Bacteremia is likely' than 'Bacteremia is 0.7 percent certain'.

2.3.4.2 Control Mechanisms

There are two primary problems addressed by the control portion of the inference engine. Firstly, the knowledge system must have a way to start. Rules and facts must be set up in a static knowledge base. Secondly, the inference engine must resolve conflicts that occur when alternative lines of reasoning emerge. In a rule-based expert system, there are several methods of performing this task of deducing conclusions from user information matched together with knowledge contained in the knowledge base. The most common mechanisms are backward-chaining and forward-chaining.

In backward-chaining, the inference mechanism guesses at a conclusion and then attempts to establish the presence of the conditions necessary to support that guess. This is the type of reasoning that humans generally employ in a diagnostic setting.
In forward-chaining, the inference mechanism compares the information in the global data base with the conditional part of a rule in the knowledge base. If the comparison reveals a match, then the rule "fires" and the result part of the rule is added to the global data base. The process repeats until no matches occur.

Reasoning in a forward chaining system is described as a "recognise-act" cycle. Firstly, the rules that can succeed, given the contents of working memory, are recognised. One rule is selected, and then the action or conclusion is asserted into working memory. The system then proceeds to the next cycle and checks again to see what rules succeed.

The inference engine is the heart of the expert system. It controls the execution or firing of rules leading to a conclusion. In this module, the contents of the global database are matched against the contents of the rule base. Rules matching the elements of the database are fired.

During the execution sequence of an expert system, more than a single rule may be applicable. In this case, a conflict resolution strategy must be used to determine the rule to be fired first. These strategies can include the following :-

1) a rule is not allowed to fire more than once on the same data (refraction).

2) rules using more recent data are preferred to rules which match against data which has been in the global database for a longer time (recency).

3) rules with a greater number of antecedents are preferred to more general rules (specificity).

While the task of selecting an inference mechanism is largely related to the choice of software for developing the system, it is evident by the nature of the project that the solution is more suited to a forward-chaining mechanism or is data-driven.

2.3.4.3 User Interface

Explanation is one aspect of user interfaces for expert systems. The ability of a system to explain how it reached a decision is crucial to making it useable. The system needs an ability to justify and explain the advice given for the following reasons :-

1) due to limited knowledge bases of expert systems, the user may want to know if the system took into account all the knowledge that the user may consider relevant.

2) the user may want to know if the strategies adopted by the system for solving the problem are satisfactory.

3) the user may wish to know if all the relevant data describing the problem state are being considered. The explanation facility of expert system is particularly important if the users are skeptical of the advice offered by the system or if the stakes involved in accepting the system's recommendations are particularly high. For example, no conscientious medical officer would accept a conclusion or prescription produced by

an expert system if the doctor did not understand and agree with the reasoning the system used to reach its conclusion.

2.3.4.4 Comparison with Conventional Programming

Although an expert system is developed and run on a computer, several important differences exist between expert systems and conventional programs. A typical conventional program is run on a complete set of data with the expectation of a unique solution to the problem. In contrast, an expert system runs on an incomplete set of data and may well produce many solutions to a problem, each with varying degrees of confidence.

Assuming a correctly written program, the outcome or result produced by the conventional program is certain; the recommendations produced by expert systems may not be. Results from the experts system may only be recommendations with associated levels of certainty phrased as "you might have this result" or "you'll probably have this result". Similarly, most expert systems include a facility for determining and displaying their confidence in the advice they offer. In conventional programs, this feature is neither present nor important.

Finally, expert systems are distinguished from conventional programs by their method of development. Most software engineers traditionally have opted for a top-down approach to software development. The project is broken down into small components, each of which may in turn be further modularised. This approach requires a view of the overall structure of the problem and an awareness of the relationships among the various modules.

In expert systems development, this overall vision is often blurred, and it is not until a certain level of knowledge has been added to the system that the structure of that knowledge becomes evident. For this reason, the top-down approach to expert systems development is often less effective. A summary of the basic distinctions between conventional programs and expert systems is outlined in Figure 6.

CONVENTIONAL PROGRAM	EXPERT SYSTEM
Requires complete set of data	Can function with incomplete set of data
Uses algorithms	Uses heuristics
Produces unique solution	May produce several solutions
Generates results that are certain	May produce uncertain results
Suitable for top-down development methodology	Suitable for bottom-up development methodology

Figure 6 : Comparison of Techniques

2.3.4.5 Programming Languages and Expert Systems Shells

In the early years of expert system research, the need for a non-procedural programming language was expressed. Many AI applications use a list processing language, LISP, developed at the Massachusetts Institute of Technology [WINSTON81]. LISP, however, is still a language in which the programmer expresses how to do things. In this concern, it is similar to conventional languages, such as FORTRAN and BASIC, even though it is much more expressive.

PROLOG (PROgramming in LOGic), a language developed in the early 1970's based on the theories of Kowalski [KOWALSKI79], is based on logic. As stated previously, mathematical logic is the fundamental basis upon which artificial intelligence and expert systems are built. PROLOG, therefore, lends itself as a natural language for building expert systems, with its structuring into clauses of facts rules and questions.

The inference engine and the user interface are often viewed as one module, a program called the expert system shell. The shell can be developed independently from the knowledge base and it represents a generic expert system containing only the general problem solving knowledge in its knowledge base. Upon this, one can build expert systems for different knowledge domains, using a predefined type of knowledge representation. Many different shells can be purchased today, significantly simplifying the process of building an expert system.

Expert system shells usually feature a rule editor for building and debugging the knowledge base and an inference engine for forward and/or backward chaining. Almost all tools include a why/how utility.

More sophisticated shells feature different knowledge representation schemes such as object description, combinations of frames and rules, graphical interfaces and various degrees of natural language processing capabilities. These features are very helpful in developing the system and understanding its reasoning processes and very important in promoting user acceptance.

A very important feature of expert system shells is the interfacing capability between the system and external databases or programs. Most well accepted shells have dedicated interfaces for full integration of database management systems and links to several programming languages.

2.3.4.6 Introduction to OPS5

The programming language OPS5 was chosen to develop the expert system. Several reasons led to the selection of this language :-

1) The problem in hand, a data-driven search space where numerous solutions were possible, and

2) OPS5 was available in several forms on a number of hardware platforms such as ExperOPS on Apple Macintosh and OPS5 running under UNIX and VMS).

OPS5 is a powerful pattern-matching language which, since its development at Carnegie-Mellon in the late 1970's, has been used to develop several large, knowledge-based systems. No rule-based language to date has been put to more rigorous use. In fact, two OPS5 systems XCON and XSEL, are in active commercial use performing complex tasks daily at Digital Equipment Corporation.

OPS5 is a production system programming language, specifically designed to test Newell and Simon's hypothesis [NEWELL72] that production rules are sufficient to explain most human cognitive behaviour.

The language is difficult to classify. By one analysis, it is a very general programming environment, a hybrid system building tool. On the other hand, it is generally been used as a production rule, forward-chaining system; and, thus, it tends to be classified as a narrowly focused tool that can aid a developer in building rule-based, forward-chaining systems.

Facts in OPS5 are represented as objects with attributes and values. Rules or productions are represented as IF-THEN statements. That is essentially all there is to the language. It has a few generic constructs that can be applied a variety of ways.

The inference engine for OPS5 is also very simple. The major event in the inference process is the "recognise-act" cycle. Rules are compared to the elements in working memory until a rule fires and new information is placed in working memory. Then the cycle begins again.

There is a simple conflict resolution scheme, which works as follows. Facts exist in an explicit working memory. When the program is started up, there is nothing in working memory. Facts are asserted directly into working memory. The basic inferencing scheme is forward-chaining, and so each IF clause is checked against working memory. If memory contains an attribute and value recognised by the IF portion of the rule, then the rule is set to fire.

All rules that might fire are collected as a group and then evaluated with a conflict resolution scheme. The essence of the resolution scheme is that rules that have not fired recently are favoured over rules that have fired recently. This is accomplished via time tags on facts in working memory. One rule will emerge as a favoured rule, and it will be fired. The actions specified by the rule will be taken, and the cycle continues, all of the IF portions of the rule set once again being tested against the contents of working memory.

CHAPTER 3

PROBLEM DOMAIN

3.1 Introduction

A number of patients, during the course of an acute or chronic illness or trauma, are unable to ingest or absorb sufficient nourishment via the oral route to promote healing, to maintain their normal body weight and body composition or to mount an immune response to infection. Thus, the incidence of medical complications increases as they become progressively malnourished.

In the last few years, significant advances have been made in the field of clinical nutrition, specifically the elucidation of many of man's specific nutritional requirements following surgery, trauma and thermal burns together with the formulation of a variety of nutritional solutions containing protein, carbohydrate, fat, minerals, trace elements and vitamins to meet these requirements and the development of safe and effective methods for delivering these nutritional solutions via the gastrointestinal tract (enteral nutrition) or the central venous route (parenteral nutrition).

Critically ill patients, usually found in the intensive care unit (ICU) of a hospital, must often be supplied with all essential nutrients. In these patients not only is nutrition vital but manipulation of internal body chemistry is one of the central concepts of intensive care therapy.

As well as being supplied with essential nutrition, critically ill patients must also have varying fluid and electrolyte physiology, which must be compensated for or treated with drugs. At any time, due to the results of laboratory tests or clinical monitoring, components of the solutions may have to be changed without altering the effect on other components within that solution.

3.2 Clinical Procedures

The appropriateness of total paranteral nutrition therapy for a particular patient is a medical decision made by a doctor as part of the patient's overall management. A patient may be assessed to require either supplementary parenteral nutrition or total parenteral nutrition by a senior medical officer according to a number of set criteria.

Following an appropriate protocol, the following information is gathered and recorded :-

1) Patient identification details (name, address, date of birth etc.)

2) Diagnosis and indications for parenteral nutrition (general assessment of state of health of patient). At this evaluation, past history and length of time without food are considered, the patient's overall state of nutrition is evaluated on fairly subjective grounds, but occasionally these may be supported by the measurement of serum albumin (a naturally occuring protein), total lymphocyte count (blood cells influencing immunity to infection), weight, weight loss, skin-fold measurement to confirm body fat content and mid-arm muscle circumference size together with weight to calculate height and surface area. Biochemical data is also considered in detail from the normal blood analysis. The final step in the sequence is to assess the patient's basic medical status. The common groupings are septic (major infection), stable post-operative, starved, pre-operative or any combination of these. Patients may also have underlying medical conditions, particularly diabetes or renal failure which will affect the basic formulation of the TPN solution.

When all the required information is available and reviewed, then the formulation can be requested either as a standard solution, influenced by the current biochemical data, or, if possible, with reference to earlier biochemical data recorded for the patient. For most patients, the former method is the one usually followed. The prescription order is written out manually and sent to the hospital pharmacy for production.

3.3 Pharmacy Procedures

The next step is the production of a formulation of base solutions and additives, in millilitres, for the pharmacist to follow whilst manufacturing the TPN solutions for the patient. A label with suitable patient identification data, and the final components of the solution in grammes, kilocalories and millimoles is then produced to be affixed to the finished product.

Dispensing hyperalimentation orders in a timely and correct manner can be very demanding for staff pharmacists. Using the computer to calculate the amounts of amino acid, dextrose and electrolyte solutions can greatly reduce the pharmacy response time and ensure greater accuracy in filling the physician's order. Having been freed from the task of manually balancing the formula, the pharmacist can spend more time checking the order for incompleteness and appropriateness.

Normally, the physician's TPN order requests the total amount of each electrolyte required along with the desired percent amino acid and dextrose. The formula is then based on the following three assumptions :-

1) The total volume is to approximate two litres.

2) The physician may order whatever percentage amino acid solution desired, as long as it is for a volume of 500 millilitres.

3) The physician may order whatever percentage dextrose solution desired, as long as it is for a volume of 500 millilitres.

The following major problems exist with the use of this technique :-

1) Unless the pharmacy, where the solutions are made up, is manned 24-hours a day by a trained pharmacist, (this does not commonly happen in practice except in large metropolitan teaching hospitals) then alterations to any of the components of a solution in the light of clinical monitoring may have to be made by the ICU staff who are not fully trained in

pharmaceutical practice or ICU staff must wait for the pharmacist on-call to arrive to prepare the solution.

2) The registrar (senior medical officer) has the legal authority to alter solutions but whilst authority to alter solutions may be delegated, in circumstances in which he is not always immediately available, his legal authority cannot be delegated.

3) Mixing techniques in the phramacy requires very strict adherence to written detail which is only learnt by practice. The order of addition together with amounts of ingredients must be closely controlled and taken into account when ordering. On average, it takes about four weeks for a resident medical officer (RMO) to become familiar with the reqirements and RMO's are replaced on a rotation basis every ten weeks.

4) TPN solutions are very expensive and have a shelf life when mixed of less than 36 hours.

3.4 Basic Requirements of the Expert System

The expert system was required as one that allows resident medical officers with limited experience to evaluate and classify patients according to set options and subsequently prescribe total parenteral nutrition solutions for patients on intravenous feeding. The prescription or order would be described in grammes of protein, kilocalories of carbohydrate or fat, and millimoles of electrolyte solution. The doctor, having classified the patient, would then be given an option to impose either fixed amounts of various components available, or broad specifications such as low sodium on the formulation of the TPN solution.

During the next stage, the program should system should prompt the doctor to enter the current biochemical profile. A more practical step would be to accept the results from an on-line transfer of biochemical data of the patient. Then the formulation could be made up either to a predefined standard solution, influenced by the current biochemical data or, if possible, with reference to earlier biochemical data which was stored within the computer system.

The system then needs to produce a written prescription order for the doctor to verify or amend and then to sign as a written script for the pharmacy. Following this is the production of the formulation of base solutions and additives for the pharmacist to follow whilst manufacturing the TPN solution for the patient, taking into account predefined safe levels of prescribing have not been exceeded, checking the availability and validity of the ingredients calculated, precipitation problems when various components are added in excess of their solubilities, drug compatibility and interactions.

The system was required to allow an experienced or authorised user to alter the preset values for standard formulations and add or alter the responses to fixed input medical situations at the initiation of a service request.

3.5 Order Sequence

The TPN order sequence consists of a number of sequential branching input requests, which, when completed, guarantee that the intent of the order has been uniquely and completely captured. When the sequence is completed, the system responds by displaying the order and requests user verification. At this time, the user may verify the order as correct, or may utilise a correction function and re-enter one or any number of the items within the order and return to the verification request. Eventually, the user must verify each order. During the process of capturing the order, the system executes numerous checks upon content of the order. The checks include maximum and minimum daily doses (per kilogram of body weight) of each medication, maximum fluid rate, potential drug-drug and drug-fluid incompatibilities.

The above description of the tasks carried out by both the ICU and Pharmacy staff shows that the decision-making process involved is complex and many heuristics are involved. This led to the belief that expert system techniques, rather than the algorithmic approach, would be suited to the implementation of a systems to advise and assist both groups. The ability to perform TPN selection, ordering and administering by medical staff is learnt by formal, on-the-job training and experience.

3.6 Specific Tools Available

A number of requirements and restrictions which existed in the program development area are listed to explain the final selection of the development tools used during the lifetime of the project.

1. The selection of programming languages available were limited to assembler (MACRO-32), COBOL, FORTRAN, BASIC on a VAX super-minicomputer and BASIC, PASCAL, C, LISP, PROLOG,...on an MS-DOS microcomputer.

2. There exists a requirement to support multiple users, separated across public boundaries, which rules out the ultimate use of small microcomputers.

3. Interpretive languages such as LISP cannot be supported on a production VAX system. The suppliers indicate that a limit of six to eight LISP users are recommended on the machines of the size in question which is required to support sixty users.

5. Indications are that most diagnosis knowledge system building tools will represent knowledge by IF-THEN rules and facts and it will incorporate a chaining inference strategy which could be well represented using conventional languages available.

6. In a non-teaching hospital environment, a small system, operational within a period of six months is mandatory. An on-going project which lasts for five years the size of MYCIN cannot be supported.

Initially, VAX BASIC was chosen due to the availability of other software development tools and the compatability of hardware for future stages of the project. OPS5 was then chosen to develop the central controlling function of the system and

VAX BASIC used to perform the conventional user and file interface functions and the more complex arithmetic.

CHAPTER 4

EXPERT SYSTEM IMPLEMENTATION

4.1 Introduction

Once it has been decided to build an expert system as a problem solving tool, one has to think about an effective and efficient implementation in the user's application environment. Some of the points that were considered during this evaluation period of the project were :-

1. the extent and complexity of the problem.
2. the time and money available for the project.
3. the existing hardware and software already in the application environment.

4. the tools available for the implementation.

Normally, choosing a software tool is not just a question of selecting a convenient piece of software, or at least it should not be. The tool to be used usually dictates the knowledge representation system to be used, and so a decision about knowledge representation had to be made first.

4.2 Implementation Environment

In the beginning of the project, the building of a system using conventional programming languages was considered and attempted. This plan, however, was discarded after it proved to be far too time consuming and not efficient or flexible enough to develop a prototype from scratch.

Initially, in the prototyping stage, the rules were written into the BASIC program code as complex nested IF-THEN-ELSE statements. This technique used in the initial prototype highlighted some problems :-

1) Low Flexibility - Having the rules permanently written into the program code greatly reduces the ability to easily and quickly modify the rules.

2) Thresholds - There exist numerous threshold limits within the deterministic rules. For example, thresholds exist for a serious result, defining some abnormality in the data and the reliability of the result. It was difficult to incorporate the use of confidence levels.

3) Contradictions - Sometimes a situation was reached which could not be resolved by the system. This requirement for human decision making highlighted the on-going problem of the growing domain, since all situations which may present themselves had not as yet been extracted from the expert and configured within the knowledge base.

The prototype written in Digital Equipment VAX BASIC was developed to a point that indicated that a computer-based solution to the TPN pharmacy calculations was in fact possible and gave valuable insights into the mechanisms and data structures that could be used in developing the expert system using a language based on a production-rule interpreter such as OPS5. It was then decided that the system would be best designed as a central OPS5 core module supported by a series of dedicated interface routines written in conventional programming languages such as VAX

BASIC or C which were added to enhance the expert system core module. The opportunity to develop a practical example using an AI language such as OPS5 was also seen to be an acceptable project from the University of Wollongong perspective.

The Illawarra Area Health Service has an information systems strategy based on a series of Digital Equipment Corporation VAX minicomputers. Both the Pharmacy and Intensive Care Unit (ICU) computer-based management systems were already implemented on VAX computers within this common resource. Since the interaction with the Pharmacy and ICU Departments was a major concern of the project, it was decided to use a programming environment running on the same host, that is, develop the system for the VAX machines.

While expert systems shells for PC's can be purchased for a few hundred dollars, the price of minicomputer programming languages and expert system shells start at the tens of thousands of dollars, not considered justifiable by the Illawarra Area Health Service considering that the project was the first of its kind within the health service. Since one of the objectives was to determine whether further work should be carried out in this area of study and whether an expert system developed using an expert system shell or AI programming language, the project had to be done on a low budget.

Considering these factors, the following solution was chosen. The system would be built using OPS5, an AI programming language available for VAX-based VMS, UNIX and microcomputers running MS/PC-DOS. A brief description of the language is given in a section later in this chapter.

4.3 System Design

The production system approach was chosen for the construction of the expert system. The principal reasons for this were :-

1) it was the most familiar approach.

2) had recently acquired OPS5 implementation and were looking for an opportunity to construct an expert system using OPS5.

3) preliminary discussions with the domain experts within the ICU and Pharmacy suggested that forward chaining would be expected to be a major control mechanism within the expert system.

The knowledge of the system is represented by rules. The rules represent relationships and are used with either object-value or object-attribute-value representations. The premise of the rule is called the expression or an if clause. The conclusion contains a single expression or then clause, although it may just as well contain more than one. The clauses in the premise are connected with logical operators.

As the expert system was developed, it became clear that OPS5 was not suitable for certain aspects of the implementation. One such aspect concerned the maintenance of all key databases, the amino acid details (containing the elemental components in each supplied batch of currently stocked pre-mixed amino acid solutions), electrolyte components (containing elemental components of each supplied batch of currently stocked electrolyte solutions), order details (containing past and current order details and completed TPN formulations) and patient details.

Originally, most of the relevant data was loaded into the global database directly, and while this kept the initialisation of the knowledge base simple, there existed a requirement to read in this data from externally maintained database files and to be able to maintain these databases efficiently and easily. The patient details resided on another central database and the solution batch components changed regularly, according to the availability of supply. The file handling facilities offered by OPS5 are primitive and not suited to operate in such an environment. Furthermore, it was planned to implement a second stage of the project, the physician's ordering module, to complement the pharmacy calculation module. The data setup during the execution of this module was complex and relatively large and required to be used for written reports, more suited to a conventional language environment.

It was therefore decided that another language would have to be used. Since the original was written in VAX BASIC, it was decided to write the database access facilities in BASIC. Other aspects of the implementation that proved inconvenient in OPS5 concerned the calculation of weights and volumes of ingredients and a user interface that would be readily accepted by staff required to use the system. The calculations were implemented in BASIC and the user interface was implemented in BASIC and TDMS (a form-based screen management system).

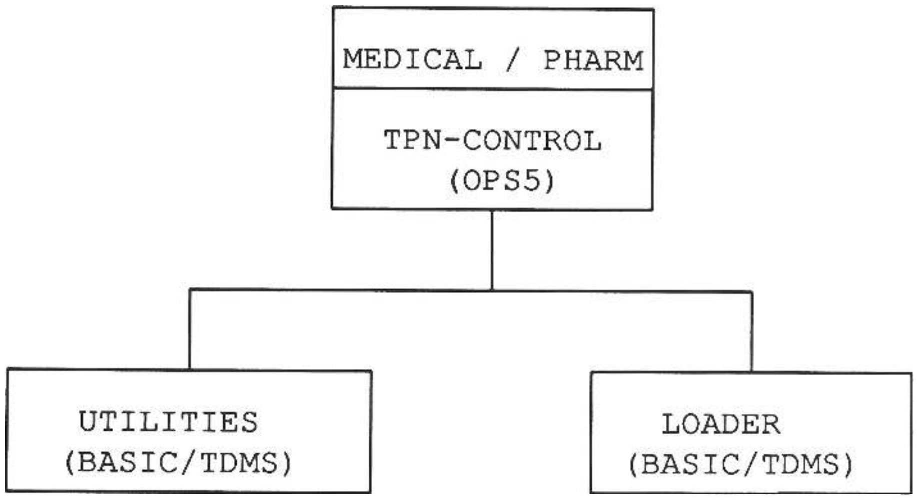


Figure 7 : Expert System Modules

The overall structure of the expert system is shown in Figure 7. The diagram depicts the major modules and their interconnections. It also contains an indication of the language in which each of the modules is written.

The TPN-CONTROL module, the main module handles user interaction and controls the overall action of the system. The module LOADER is concerned with the maintenance of the databases mentioned above. The UTILITIES module consists of routines used by TPN-CONTROL to retrieve records describing a patient, electrolyte components, user interaction and selected calculations considered to be inconvenient and overly complex when performed in OPS5.

As shown in the diagram above, TPN-CONTROL is written in OPS5, whereas UTILITIES and the LOADER modules are written in BASIC and TDMS. The figure also shows the way in which the BASIC routines are subordinate to the OPS5 modules. The TPN-CONTROL module is the top level of the system and it calls routines in the other two modules, with a negligible amount of interaction between the BASIC modules.

The medical and pharmacy sections of the TPN-CONTROL module contain a mixture of predefined and volatile data, kept in permanent disk files and entered through the keyboard respectively. For example, a 'stable starved' patient category is calculated from the weight loss (less than 10% over four to eight weeks), assessed appearance (wasted, qualified by determining mid-arm muscle circumference), biochemistry (normal with low serum urea) and haematology (folate or B12 deficiency).

Based on the patient's category, an initial solution prescription of protein, calories, electrolytes and other ingredients (insulin and vitamins) is calculated and a TPN order produced. The system data relationships appear as follows :-

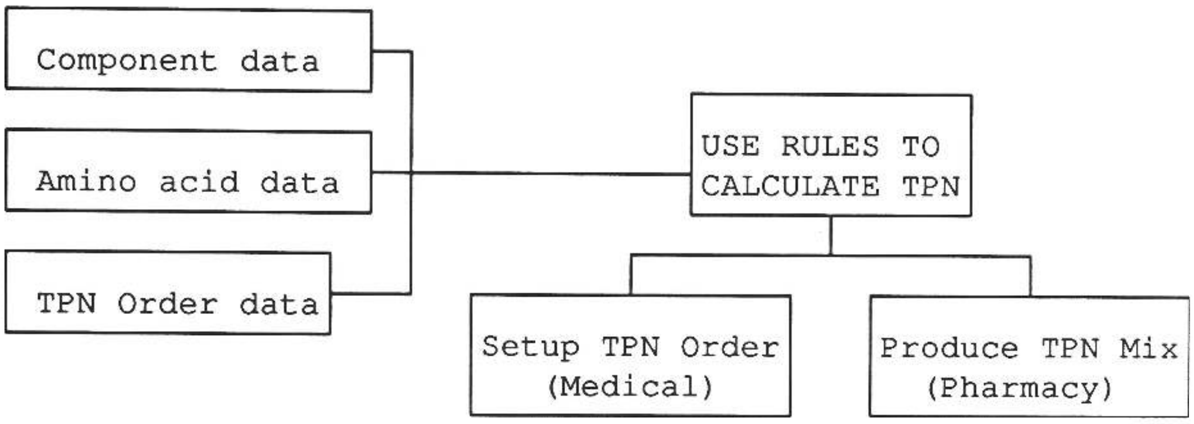

Component data
Amino acid data
TPN Order data
USE RULES TO CALCULATE TPN
Setup TPN Order (Medical)
Produce TPN Mix (Pharmacy)

Figure 8 : Data Relationships

4.4 The OPS5 Language

The OPS5 language, invented by Charles Forgy, John McDermott, Allen Newell and Mike Rychener at Carnegie Mellon University during the late 1970s, is one of the best known languages for the development of rule-based expert systems. OPS5 and the earlier OPS languages were not originally intended for expert system development, but were used in solving other AI problems, in solutions that used production systems, such as cognitive modelling for research in psychology and cognitive science. However, since John McDermott used OPS5 to build the highly successful R1 (also know as XCON) expert system for configuring Digital Equipment VAX minicomputers, the language has received considerable attention as one of the better known rule-based systems.

OPS5 is a modest language, offering a small set of features and limited syntax. The original development of the language came from the development of a very fast pattern matching algorithm, called Rete [FORGY82]. This algorithm was developed to offer an efficient method of comparing a large collection of patterns to a large collection of objects, for use in a production system - an unordered collection of if-then statements commonly called productions.

Generally, OPS5 uses a data-driven, forward-chaining inference engine. This type of reasoning is appropriate when it is possible to provide the system with all the data initially so that it can apply as many rules as are appropriate to make inferences on the way to a solution.

The OPS5 language can be best described as bundles of techniques and mechanisms for developing production systems, rather than as a fixed shell for building a particular type of expert system. The control mechanism is a basic loop that establishes a "recognise-act" cycle. Rules can be used to implement different control strategies involving such a loop. The loop runs as follows :

1) **Match:** Find all the rules whose antecedent conditions are satisfied by the known initial data.

2) **Conflict Resolution:** Select one of the rules with satisfied antecedent conditions. If there are none, then exit.

3) **Act:** Execute the action prescribed in the selected rule's result clause.

4) **Repeat:** Return to step 1.

XCON is implemented as a production system. It uses the "match" as a principal problem solving method. It has sufficient knowledge of the configuration domain and of the peculiarities of the various configuration constraints that exist at each step of the configuration process and it simply recognises what to do. Consequently, little search is required in order for it to configure a computer system.

The forward-chaining solution suits the problem in-hand since there exists a large number of data in the initial state from which a solution must be found and there is typically no single nor optimal goal state. There is only a set of constraints to which the goal must conform. In simple terms, there is an initial state of many facts that must be synthesised into a solution.

4.4.1 Use of atoms, classes and attributes in OPS5

A program in OPS5 consists of a declaration section that describes the data objects of the program, followed by the production section that contains the rules. During execution, data operated on by the program are kept in working memory and the rules in production memory.

Working memory is initialised after the declarations and rules have been loaded. The declaration section contains the definitions of the data object types and of all the user-defined functions that are to be referenced in the rules.

The OPS5 language supports two primitive data types, numeric and symbolic atoms. The numeric atom may contain integer or floating-point values and a symbolic atom is any sequence of characters that is not a number. A sequence is a group of characters that may be treated as a single unit.

The compound data structure type in an OPS5 program is an element class. It is similar to a structure declaration in C or COBOL, or a record declaration in BASIC or Pascal. The components of an element class are called attributes.

Since OPS5 is not a strongly typed language, the declarations do not include a type specification for the values of the attributes of the element class. An element class, C, is declared as follows :-

```
(literalize C
            attribute1
            attribute2
            ...
            attributeN)
```

where C is the class name and attribute1,...,attributeN are the attribute names and are both symbolic atoms.

4.5 Knowledge Base

The expert knowledge of the physician and the pharmacist is principally contained within the TPN-CONTROL module. This knowledge was obtained from written protocols, policy and procedural documents and through many hours of observation and questioning.

Both medical and pharmacy staff were very co-operative, motivated at least in part by the belief that it would not be possible to place their hard-won expertise into a computer program. The only real difficulties with the knowledge acquisition aspect of the project were :-

1) Both the medical and pharmacy environments sometimes required deep understanding of many associated topics to realise answers to only a small problem area.

2) In some cases it was difficult for the staff to explain the fundamental reasoning why a decision was made or were not aware of all the factors that they took into account in coming to the decision.

3) The rapid turnover of staff in the medical area.

The last problem was a major hurdle to the project from the onset, since the knowledge base was required to be completely rebuilt several times to follow fundamentally different approaches to the problem.

4.5.1 TPN Medical Module

When a new TPN order is requested to be entered or an existing order requested to be updated, all relevant information about the patient is loaded into the working memory of the production system. The productions are then applied, gradually building up a patient profile and adjusting the ingredients required to meet with the requirements of that profile under controlled conditions until a suitable a final prescription order is produced. At the end of the process, the best formulation is presented to the physician to be accepted or modified. The order may then be printed and released by the physician and become a formal script to be dispensed by the Pharmacy.

4.5.2 TPN Pharmacy Module

As in the medical module, when a new TPN order is requested to be entered or an existing order requested to be updated, all relevant information about the patient is pushed into the working memory of the production system. This module takes the list of requirements in terms of groups of additions and calculates the combinations of amounts of ingredients required to produce a correct mix of amino acids, fats, carbohydrate, vitamins, minerals and trace elements within the volume of solution required.

The pharmacy staff may use this module with or without the medical module, since not all TPN formulation orders come from the ICU. Other recipients of TPN manufacture are the general wards, maternity and special care nurseries and patients offered specialist home care. In this case, the pharmacy staff make up the order depending on the medical staff instructions and then calculate the formulation.

4.5.3 Data Structures

The working memory of the production system within the TPN-CONTROL module contains several elements or classes including :-

1) ORDER_DETAILS, which describes the medical and pharmacy related information for each order,

2) PATIENT_DETAILS, which describes the patient specific information,

3) BATCH_DETAILS, which describes the addition substance and amount for each TPN batch produced,

4) AMINO_ACID_DETAILS and COMPONENT_DETAILS, which contain the elemental component amounts for these ingredients,

5) ACTIVITY, which is used to control the order in which some of the productions are permitted to fire,

6) EXPLANATION_DETAILS, which is used to give a traceback of how the current rule firings were achieved.

These working memory elements frequently have internal structure, in that they consist of attribute-value pairs. For example, the class called ORDER_DETAILS contains attributes for the patient medical record number, the date that the order was requested, the date that the TPN order was dispensed by the pharmacy and details relating to the ingredients that make up the TPN prescription.

The productions in the TPN-CONTROL module are predominantly concerned with the examination of the factors described in the previous chapter. One of these productions from the medical section is shown below :-

```
(p check_patient_type
      { <This_activity>
       (activity ^taskname get_patient_type_task ) }
      { <This_patient_order>
      (order_details ^mrn <entered_mrn> ^category nil) }
-->
      (modify <This_activity> ^taskname setup_order)
      (modify <This_patient_order>
                  ^category (get_patient_type)))
```

Figure 9 : Example Production

An OPS5 production consists of a '(p', the name of the production, the left-hand side of the production, the '-->' symbol, the right-hand of the production and a closing ')'. The left-hand side of a production is a collection of patterns called condition elements, each of which is a pattern to match a working memory element. For example, the first condition element in the example of code above is :-

```
{ <This_activity>
(activity ^taskname get_patient_type_task) }
```

and this condition element evaluates to 'true' if the working memory contains an element 'activity' and the value of its second attribute, 'taskname' is 'get_patient_type_task'.

```
{ <This_patient_order>
(order_details ^mrn <entered_mrn> ^category nil) }
```

The second condition element, 'order_details' succeeds if the working memory contains an element denoted by the class variable '<entered_mrn>' with a second attribute called 'category' which has an empty value; i.e., it contains the symbolic atom 'nil' set up when the attribute was initialised. When the symbolic atom 'nil' is used to describe the intended value of an attribute, it means that nothing is known about that attribute.

The right-hand side of an OPS5 production consists of an sequence of commands called actions. There are two such actions shown in the example of code in Figure 9.

The first right-hand side action is :-

```
(modify <This_activity> ^taskname setup_order )
```

The first action consists of a modify statement. The modify action is used to change specific attributes of a particular element of working memory and update the working-memory element unique integer identifier or time tag. The command takes the first pattern on the left-hand side of the production associated with the element variable '<This_activity>' and creates a copy of the working memory element whose contents are identical as the element associated with the element variable '<This_activity>', except that the second attribute now contains the value 'setup_order'. The previous element associated with this name is then removed from working memory.

As an alternative to the modify command, one could remove the element first and then make a new element. However, usually only a few of the attribute values are to be changed, and it is inefficient and cumbersome to have to know and explicitly represent all the other attributes and values for the use of the make action.

The second action is :-

```
(modify <This_patient_order>
    ^category (get_patient_type)) )
```

This action calls an external function written in BASIC called 'get_patient_type' to modify the specific order_details working-memory element. The external function prompts the user for information about the patient so that a patient profile can be created. The function returns the patient category.

The effect of the production described above is to test whether the TPN order under consideration is for a patient who has been assessed and assigned a patient type. If the patient has not been assigned a patient type then request the user for details to make an assessment and record it in the TPN order data structure for further reference.

Both the medical and pharmacy modules use the ACTIVITY data structure for a context-limiting and specificity-ordering conflict resolution strategy based on techniques incorporated into XCON. The context-limiting is done by making the first clause in each rule conditional on what is currently being performed by the program. Specificity ordering enables the module to use rules such as the following for switching contexts :-

```
RULE XYZ:  if the current context (activity) is X
           then     deactivate the X context
                    activate the Y context.
```

This rule mechanism has the effect of deleting one item from the context designation and then adding another. It fires only if no other rule associated with the context triggers, for any other triggering rule would have conditions that are a superset of unaccompanied context checking.

The EXPLANATION_DETAILS data structure holds a list of rule firings. By looking into a historical record, a rule-based system can answer questions about why a rule was used or how a fact was established. Much of this ability relies upon the simple, highly constrained format that the system paradigm imposes on its rules. To decide how a given fact was concluded, for example, the system need only reflect on the rules it has used.

4.6 Communication with External Routines

As mentioned in the previous sections, the BASIC routines contained in the other modules are called by the OPS5 modules. Establishing the correct interface between the two languages was much more difficult than originally anticipated and it provides a prime example of the practical difficulties that may arise when attempting to make use of an 'expert system language' such as OPS5. Some of the difficulties encountered were due to the brevity of the OPS5 documentation and others were because of the complex and tedious nature of the interface protocol required.

```
(external bas$display_order               ;external utilities
          bas$print_label
          bas$store_order
          ... )

(literalize activity
          taskname                        ; rule cluster
          subtask                         ; sub-group
          ...)

(literalize order_details
          mrn                             ; patient medical record number
          order_date
          dispense_date
          ...
          order_status
          metabolic_status                ; patient condition
          ... )

(literalize patient_details
          mrn
          ... )
```

```
(literalize batch_details                    ;audit trail
          mrn
          ... )

(p setup_order
          { <This_activity> (activity ^taskname setup_order) }
          { <This_patient> (patient_details ^mrn <mrn> ) }
          { <This_order> (order_details ^mrn <mrn>
                                        ^order_date <date> }
-->
          (modify <This_order> ^order_date <date> )
          (make batch_details ^mrn <mrn>
                                        dispense_date <date>
                                        ... )
          (call bas$display_order)
          (call bas$print_label)
          (call bas$store_order <mrn> <date> )
          (modify <This_activity> ^taskname complete
                                        ^subtask complete ))
```

Figure 10 : Example of OPS5 code calling BASIC routines

All communication between OPS5 and the user's external function is via a set of OPS5 functions that manipulate a special entity called the result element. The result element is a template for a working memory element that has been set up by a make, modify or call command.

The result element contains a vector of fields that is identical to the WME, but that sits outside working memory. It is the only interface between the external routine and the OPS5 code, since an external routine cannot search working memory nor remove or modify WME directly. The OPS5 code fills the result element with values as arguments to send to the external routine, and the external routine fills the result element with values, if it is sending a WME back to the OPS5 program.

Routine	Argument	Purpose and Return Value
OPS$CVAN	(atom)	OPS5 convert atom to number. Converts an integer atom to an integer and returns the integer.
OPS$CVNA	(integer)	OPS5 convert number to atom. Converts an integer to an integer atom and returns the atom.
OPS$CVAF	(atom)	OPS5 convert atom to floating. Convert a floating-point atom to a floating-point number and return the floating-point number.
OPS$CVFA	(floating)	OPS5 convert floating to atom. Convert a floating-point number to a floating-point atom and return the atom.
OPS$PNAME	(atom, character buffer, buffer-size)	OPS5 print name. Fill the character buffer with the print-name of the atom and return the number of characters in the buffer.
OPS$INTERN	(character buffer size)	OPS5 internal symbol. Convert the contents of the buffer to a symbolic atom and return the symbolic atom.

Figure 11 : VAX OPS5 atom conversion routines.

Since all OPS5 values are represented as atoms, in non-LISP implementations of OPS5 such as the version used for the project, routines written in other languages cannot rely on a shared understanding of the atom, and atom conversion is necessary.

The OPS5 interface-support routines used for atom conversion that are supplied by VAX OPS5 are listed in Figure 11.

There are four commands that can be called from within the external routine to build up the result element or insert its contents into working memory. The first command, 'ops$reset', clears all the fields in the result element and sets the default insertion location to be the first field. The second command, 'ops$tab', controls where the next value will be placed. The third command, 'ops$value', puts a value into the result element of the current field and increments the pointer to the next field in the current class. The final command, 'ops$assert', copies the result element into working memory but does not change the contents of the result element. Other commands such as 'ops$intern' and 'ops$pname' are used to convert to OPS5 form and from OPS5 form respectively.

Both external functions and external subroutines can use the 'ops$value' support routine to place atoms in the result element. External functions use the 'ops$value' support routine to return a single atom back to the OPS5 calling program. External subroutines may use the routine to create new working-memory elements available to the OPS5 program. This side-effect was found to be difficult to use, often resulting in confusing and erroneous results.

The method of passing arguments to functions differs from that of subroutines. In subroutines, the arguments are placed in the result element and are retrieved one at a time, whereas in functions, the arguments listed in the function call are mapped directly into the external function's parameter list.

One further interesting aspect of function calls from an OPS5 routine is that a RHS function is expected to put a value in the result element, but it must use the OPS5 interface-support routine 'ops$value' to do so. This support routine deposits one value in the result element in the position that the function appeared in the RHS OPS5 pattern. The standard function return is ignored. This means that one has the option of writing a function that has no return value at all and makes external OPS5 functions unacceptable as standard external functions to be called by other conventional languages.

The arguments that are passed to the function and the value that a function returns must evaluate to an atom, a 32-bit longword representing an integer or a number. If a call to an external function includes arguments, those arguments must be declared external to the function's definition.

The way in which the interaction between OPS5 and BASIC must be accomplished is illustrated by the example in Figures 10 and 12. Figure 10 contains an extract of OPS5 code, consisting of three productions and associated declarations. Among these declarations is the indication that there are three external routines known as 'bas$display_order', 'bas$print_label' and 'bas$store_order'. The calls to these routines are quite simple and are shown at the end of the production known as 'setup_order'. The parameters required to be transmitted to the BASIC routine, such as '<mrn>' and '<date>' are simply included as part of the call statement to the 'bas$store_order' subroutine outlined in Figure 12.

```
1          SUB BAS$STORE_ORDER
           !
           ! Definitions of OPS5 support routines,
           !  record structures and variable declarations
           !
           ...

           ! Obtain the medical record number.
           mrn = ops$parameter(1% by value)
           ! Convert the symbolic atom into a string.
           chars% = ops$pname(mrn_location, mrn, 8%)
           ! Obtain the date and convert into a string.
           date = ops$parameter(2% by value)
           chars% = ops$pname(date_location by value, date, 6%)
           !
           ! Perform file update operations.
           ...
           !
           EXIT SUB
            Perform error handling.
           END SUB
```

Figure 12 : An outline of the BASIC subroutine 'bas$store_order'.

The BASIC side of the interface, however, reveals the tedious code which is required merely to transfer information into and out of the BASIC routine. This complexity is illustrated by the BASIC procedures 'bas$store_order' and 'bas$calculate_amino' outlined in Figures 12 and 13. The routine 'bas$store_order' is used to perform file output, thereby bypassing the primitive file handling offered by the OPS5 language and 'bas$calculate_amino' performs arithmetic which is difficult and cumbersome to code in OPS5.

```
1       LONG FUNCTION BAS$CALCULATE_AMINO (ATOM1, ATOM2)
        !
        ! Definitions of OPS5 support routines,
        !  record structures and variable declarations
        !
        ...

100     ! Obtain the required amino amount.
        if ops$floating(loc(atom1) by value)
          then
            required_amino = ops$cvaf(loc(atom1) by value)
          else
            required_amino = ops$cvan(loc(atom1) by value)
        end if
        ! Obtain the amino present in pre-mix.
        if ops$floating(loc(atom2) by value)
          then
            required_amino = ops$cvaf(loc(atom2) by value)
          else
            required_amino = ops$cvan(loc(atom2) by value)
        end if
        !
        ! Perform the calculations.
        !
        ...
        ! Convert the calculated amino amount result into
        !  a floating-point atom.
        atom% = ops$cvfa(calculated_amino_amount by value)
        ! Place an atom into the result element's first field.
        call ops$value by value (atom%)
        EXIT FUNCTION
        ! Perform error handling.
        ...
        END FUNCTION
```

Figure 13 : An outline of the BASIC function 'bas$calculate_amino'.

Incoming parameters to the BASIC routine are packaged in a result element by OPS5 and these have to be extracted using calls on 'ops$parameter'. It is also necessary to convert these parameters from a symbolic, integer or floating-point atom into the corresponding data type suitable for BASIC using a corresponding set of routines called 'ops$name', 'ops$cvan' and 'ops$cvaf'. Where the symbol type of the incoming parameter is known, the corresponding conversion may be performed. In OPS5, however, numeric symbols may be either symbolic, integer or floating-point depending on the existence of the decimal part of the number and so the boolean type predicate routines 'ops$symbol', 'ops$integer' and 'ops$floating' must be used to determine the atom type and thus perform the correct conversion. There is also corresponding routines to convert the three different data types into symbols, integer and floating-point atoms.

```
1           SUB BAS$READ_ORDER
            !
            ! Definitions of OPS5 support routines,
            !  record structures and variable declarations
            !
            ...

            ! Obtain the medical record number.
            mrn = ops$parameter(1% by value)
            ! Convert the symbolic atom into a string.
            chars% = ops$pname(mrn_location, mrn, 8%)
            ! Obtain the date and convert into a string.
            date = ops$parameter(2% by value)
            chars% = ops$pname(date_location by value, date, 6%)
            !
            ! Perform file read operations.
            ...
            !
            ! Delete the atoms in the result element.
```

```
call ops$reset()
! Convert the numeric string into a symbolic atom.
atom% = ops$intern(order::mrn by ref, 8% by value)
! Create atom for class name and put it into the
!  first field of the result element.
classname$ = "ORDER_DETAILS"
atom% = ops$intern(classname$ by ref,   &
                len(classname$) by value)
call ops$value by value (atom%)
! Convert the string mrn into a symbolic atom and
!  place it next in the result element.
fieldname$ = "MRN"
atom% = ops$intern(fieldname$ by ref,   &
                len(fieldname$) by value)
call ops$value by value (atom%)
! Convert the string date into a symbolic atom and
!  find out the position in the WME and place it
!  in the correct position in the result element.
fieldname$ = "ORDER_DATE"
atom% = ops$intern(fieldname$ by ref,   &
                len(fieldname$) by value)
call ops$tab(atom% by value)
call ops$value by value (atom%)
! Set up other fields.
...
! Copy the contents of the result element to working
!  memory and create a new WME.
call ops$assert()
!
EXIT SUB
 Perform error handling.
...
END SUB
```

Figure 14 : An outline of the BASIC procedure 'bas$read_order'.

Since OPS5 is not a strongly typed language, there is no mechanism for declaring a variable or attribute name to be a particular type or to define subtypes of the essential primitive types of number or symbolic atom. This feature added further levels of complexity to the OPS5 routines, since type mismatches were only detected at run-time and it is not clearly evident to see an error when applying relational operators to values of different types. The test simply evaluates to false.

The influence of weak data typing most difficult to detect was the external call interface to BASIC. An example of this problem is highlighted in the BASIC function 'bas$get_calculate_amino' shown in Figure 13.

A further example, showing the code necessary to return values from an external BASIC routine is shown in Figure 14. To return information, the result element is cleared using 'ops$reset' and its various fields are set correctly using 'ops$value', first performing conversions into suitable OPS5 form using a routine called 'ops$intern'. When all fields are set up, 'ops$assert' is called to create a new working-memory element with the name to be found in the first field of the result element. To return multiple results, for example to read multiple records from a file and set up working memory elements from each record, the result element must then be re-used, employing further calls on 'ops$reset', 'ops$value' and 'ops$assert' to create each working memory element.

It was found that the side-effects that an external BASIC routine has on the working-memory of an OPS5 module are frequently extensive, with the BASIC routine using 'ops$assert' to create working memory elements whose origins are unclear from an examination of the OPS5 code. For example, the working memory element created by the call to 'ops$assert' at the end of the procedure 'bas$read_order' shown in Figure 14 has the name 'order_details', due to the first calls to 'ops$intern and 'ops$value'.

This working memory element is mentioned in several condition elements on the left-hand side of productions in the OPS5 module, including the production 'order_details' in Figure 9, but never appears in the OPS5 code in a situation corresponding to the creation of a working memory element of this name, making the OPS5 module difficult to understand. The primitive but complex nature of this interface between OPS5 and BASIC greatly impairs the understanding of both modules and contributed significantly to the development time the project.

CHAPTER 5

ANNOTATED SYSTEM

5.1 Introduction

After satisfying a standard user identification and password security checks, the user is presented with the initial TPN menu offering the following selections :-

```
          ICU  /   PHARMACY
    PARENTERAL NUTRITION SYSTEM

 ┌──────────────────────────────┐
 │                              │
 │ 1. Order Patient TPN         │
 │                              │
 │ 2. Check TPN Order and Print │
 │                              │
 │ 3. Maintain Data Files       │
 │                              │
 └──────────────────────────────┘

     ┌───────────────────────┐
     │ Enter Selection :     │
     └───────────────────────┘
```

Selection 1 is designed to be used by the medical staff to assist with creating a TPN order containing the basic requirements for the TPN solution, based on the patient clinical assessment and laboratory results. A base order is displayed on the screen, for amendment, if required. The solution requirements are then stored on file and the TPN order printed.

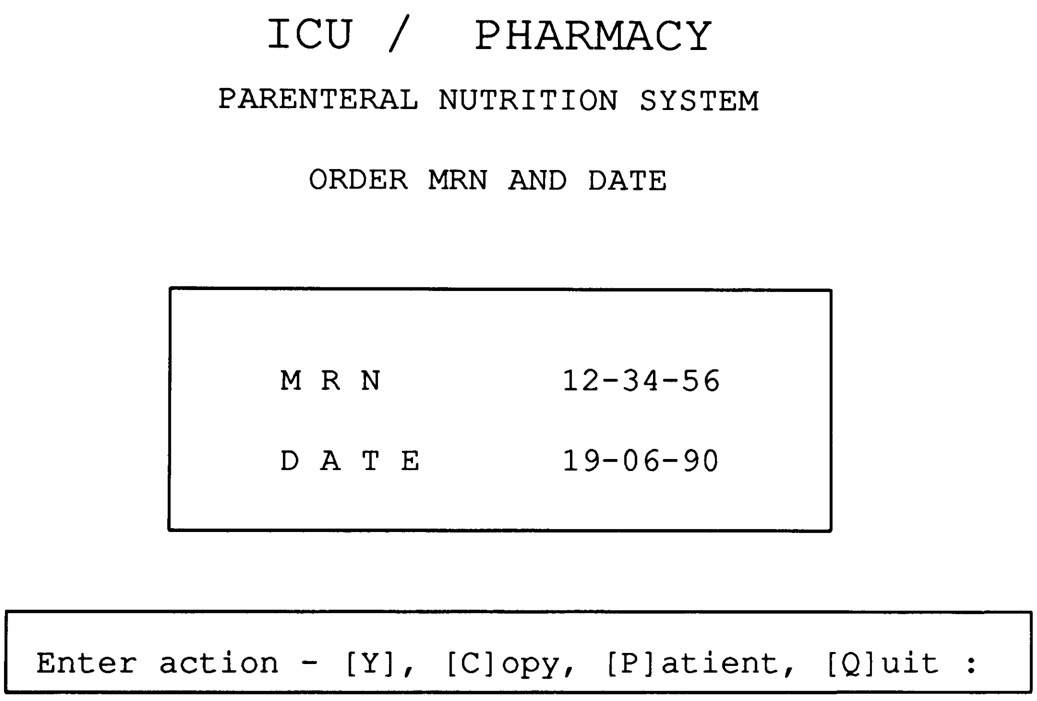
Selection 2 is designed to be used by the pharamacist to perform the necessary calculations to generate a TPN formula, given the basic order details entered by the medical staff via selection 1. Details about the components used to make up the TPN solution are retrieved from file. When the calculations are complete, the formula is displayed on the screen, showing the amounts of each element to be added. If the pharmacist accepts the calculated TPN formulation, the order may be printed onto adhesive labels, to be attached to the TPN container.

Selection 3 is used to carry out file maintenance on information relating to the composition of the amino acids and other components used in the formulation.

5.2 Medical Order

On entry to selection 1 of the main menu, the screen layout shown in Figure 15 is displayed. This screen form requests the user to enter the medical record number associated with the patient and the date of the order to be produced. The default date displayed is the current date.

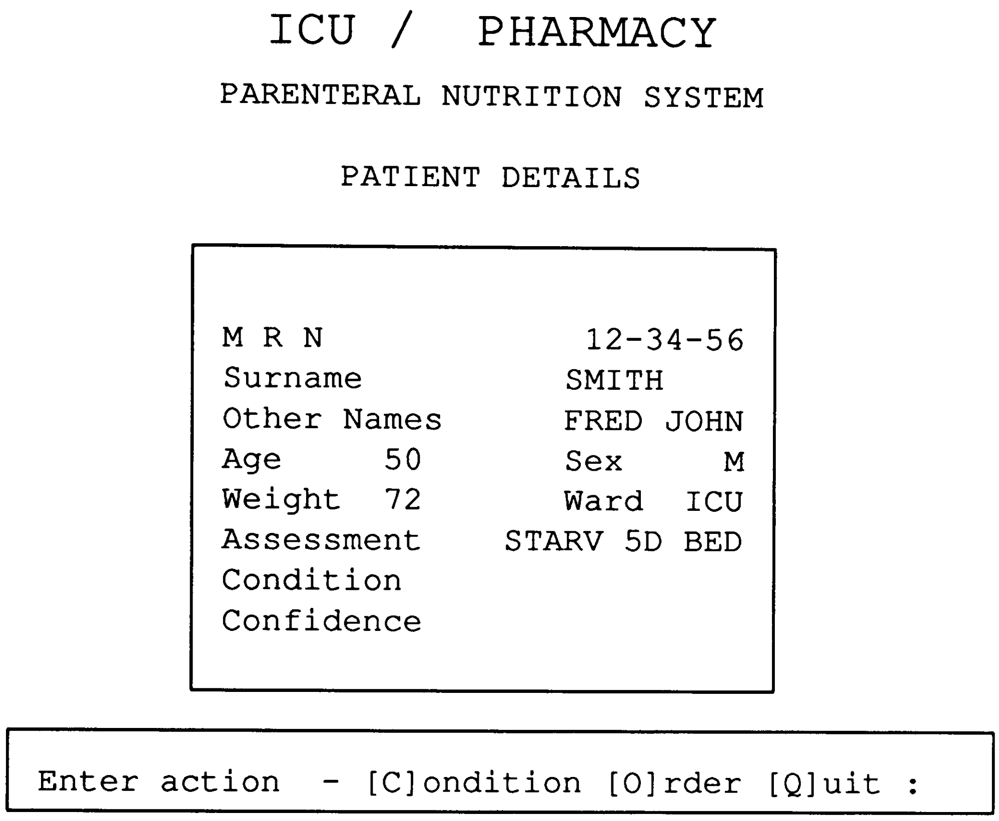
The option of selecting a specific date other than the current date is present since both medical staff and pharmacy staff have a need to use the copy command 'C' to duplicate an existing order for a patient. This is due to the fact that it is often a common situation to continue a repeating script for similar TPN mixtures as part of a patient's treatment over an extended period. In addition, the pharmacy staff are often required to prepare TPN mixtures in advance.



```
                 ICU  /  PHARMACY
           PARENTERAL NUTRITION SYSTEM
               ORDER MRN AND DATE

        M R N           12-34-56
        D A T E         19-06-90

 Enter action - [Y], [C]opy, [P]atient, [Q]uit :
```

Figure 15 : Initial Order Selection Screen

After the medical record number and the date are entered, the command 'P' for patient information is entered, and the patient screen as shown in Figure 16 is displayed.



```
                ICU  /  PHARMACY
          PARENTERAL NUTRITION SYSTEM

                PATIENT DETAILS

        M R N                12-34-56
        Surname             SMITH
        Other Names         FRED JOHN
        Age        50       Sex       M
        Weight  72          Ward   ICU
        Assessment      STARV 5D BED
        Condition
        Confidence

  Enter action   - [C]ondition [O]rder [Q]uit :
```

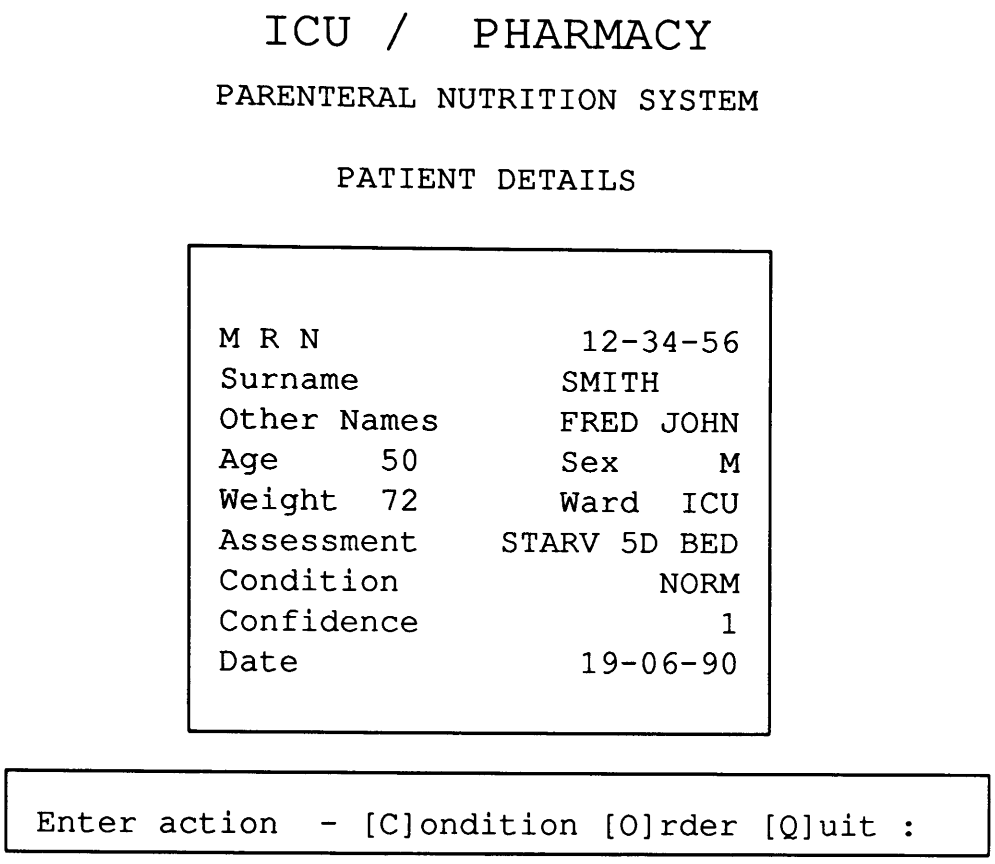
Figure 16 : Adding Assessment to Patient Screen

The medical staff may now add the patient assessment information to the patient screen. In Figure 16, the assessment text 'STARV 5D BED' has been entered by the physician. The text indicates that the patient was assessed to be starved for less than five days and that the patient mobility is static i.e., the patient is bed-ridden. Such information is processed by the rule-base, to determine initial order conditions.

When these assessment details are complete, the command 'C' is entered. The system then retrieves the relevant laboratory information for the patient from an on-line laboratory results reporting database. This information, together with the patient assessment details entered by the physician is processed by the rule-base in an attempt to make an assessment of the patient's metabolic status. The resulting screen is shown in Figure 17.

The patient condition in this example is determined to be normal i.e., stable and starved for less than five days. The laboratory results combined with the assessment details indicate a certainty factor of 1 i.e., definitely agree.

The details from a previously generated order may be copied as a basis for a new order and the patient details may be checked and amended via the patient screen shown in Figure 16 or 17. The 'C' command step may be skipped if the physician wishes to enter the patient condition directly.



```
                 ICU /  PHARMACY

          PARENTERAL NUTRITION SYSTEM

                PATIENT DETAILS

     M R N                     12-34-56
     Surname                  SMITH
     Other Names              FRED JOHN
     Age       50             Sex      M
     Weight  72               Ward   ICU
     Assessment           STARV 5D BED
     Condition                     NORM
     Confidence                       1
     Date                      19-06-90

  Enter action  - [C]ondition [O]rder [Q]uit :
```

Figure 17 : Generate Order from Patient Screen

There is considerable variance of the initial formula depending on both the patient's basic status and the current thinking of the person providing the formulation. There are two basic methods by which the formula may be provided, either as a fixed projection on the estimated average requirements or as a calculated system on patient protein turnover. Although it varies slightly, approximately 50% of the patients have a TPN formulation generated by the calculation method and the other half of the cases is often done as a teaching exercise for the resident medical staff by the ICU staff specialist.

The patient classification scheme rule-base, built on a combination of laboratory result comparisons, together with medical diagnostic rules of thumb, includes rules such as :-

```
RULE : get_metabolic_status::get_lab_results

                IF in correct context
                AND patient information exists for this mrn
                AND patient condition is defined
                AND metabolic_status is not defined

                THEN retrieve laboratory results for this mrn
                AND set context to next
```

In OPS5, the rule appears as :-

```
    (p get_metabolic_status::get_lab_results
        { <This_activity> (activity ^taskname get_metabolic_status)
        { <This_patient> (patient_details ^mrn <mrn>
                                    ^metabolic_status nil) }
        { <This_explanation> (explanation_details ^rule_counter
                                    <current_rule_count>) }
-->
    (bind <current_rule_count> compute (current_rule_count> + 1))
    (modify <This_explanation> ^rule_counter <current_rule_count>
        ^current_rule |get_metabolic_status::get_lab_results|
        ^rule_text |Retrieve lab results in order to calculate
                                    patient metabolic status| )
    (call bas$get_labresults <mrn>)    ;Retrieve lab details.
    (modify <This_activity> ^taskname calc_metabolic_status)
    (call bas$display_patient <mrn>))
```

This rule firing loads the required laboratory data into the working memory of the medical module and sets the context for selection from a series of rules such as the following, designed to calculate a possible patient classification and associated confidence factor :-

RULE : get_metabolic_status::calculate_normal

```
IF in correct context
AND patient information exists for this mrn
AND metabolic_status is not defined
AND patient condition is starved less than 5 days
AND laboratory results <= normal levels
AND laboratory results > low levels

THEN patient is NORMAL
AND confidence is DEFINITELY
AND set context to next
```

In OPS5, the rule appears as :-

```
    (p get_metabolic_status::calculate_normal
         { <This_activity> (activity ^taskname calc_metabolic_status)
; Check if status not defined and patient starved
;  for 5 or less days.
         { <This_patient> (patient_details ^mrn <mrn>
                                           ^metabolic_status nil
                                           ^starved_days { <> nil
                                                           <= 5 }) }
; Get laboratory test tolerance values.
         { <This_range> (lab_result_range ^protein_norm <pro_normal>
                                           ^protein_low <pro_low>
                                           ^albumin_norm <alb_normal>
                                           ^albumin_low <alb_low>
                                           ^platelet_norm <pl_normal>
                                           ^platelet_low <pl_low>) }
;
; Check if laboratory results are between normal levels and
; low levels for specific biochemistry and haematology tests.
;
         { <This_result> (lab_results ^mrn <mrn>
                                           ^protein { <= <pro_normal>
                                                      > <pro_low> }
                                           ^albumin { <= <alb_normal>
                                                      > <alb_low> }
                                           ^platelet { <= <pl_normal>
                                                       > <pl_low> } ) }
-->
    (modify <This_patient> (patient_details ^mrn <mrn>
                                           ^metabolic_status NORM
                                           ^confidence 1)
    (call bas$store_patient <mrn>) ;Record for screen display.
    (call bas$display_patient <mrn>)
    (modify <This_activity> ^taskname setup_required))
```

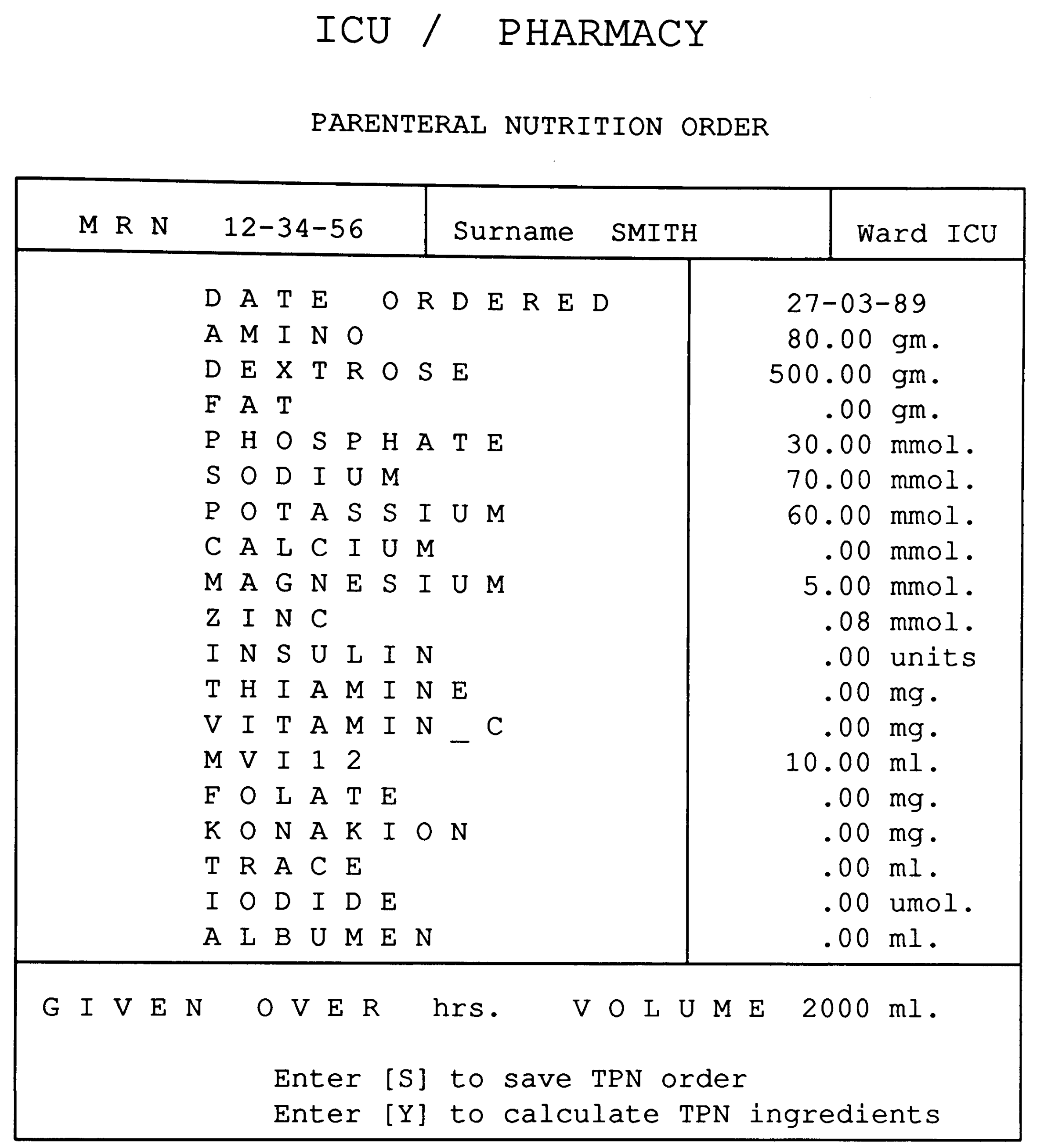
The patient may be classified into one of the following seven categories :-

CODE	CATEGORY
NORM	Normal, stable
STRES	Stressed
STARV	Starved, stable
RENAL	Renal failure
COMA	Hepatic failure
RESP	Respiratory failure
ORGAN	Multiple organ failure

The patient condition has an associated certainty factor based on the combinations of information used to calculate it. This scale of confidence consists of a discrete range of values between -1 and +1 as follows :-

VALUE	MEANING
1.0	DEFINITELY
0.8	ALMOST CERTAIN
0.6	PROBABLY
0.3	SLIGHT EVIDENCE
0.0	IGNORE
-0.6	PROBABLY NOT
-0.8	ALMOST CERTAINLY NOT
-1.0	DEFINITELY NOT

The medical staff are now required to make adjustments to the information shown on the patient screen relating to the patient's overall appearance and clinical condition. For example, the assessment text 'BED' has been entered to indicate that the patient is bed-ridden. This fact is used when calculating the energy requirements of the initial order. When this step is complete, the command 'O' generates the initial order requirements and displays the order form as shown in Figure 18.



ICU / PHARMACY

PARENTERAL NUTRITION ORDER

M R N 12-34-56	Surname SMITH	Ward ICU

D A T E O R D E R E D	27-03-89
A M I N O	80.00 gm.
D E X T R O S E	500.00 gm.
F A T	.00 gm.
P H O S P H A T E	30.00 mmol.
S O D I U M	70.00 mmol.
P O T A S S I U M	60.00 mmol.
C A L C I U M	.00 mmol.
M A G N E S I U M	5.00 mmol.
Z I N C	.08 mmol.
I N S U L I N	.00 units
T H I A M I N E	.00 mg.
V I T A M I N _ C	.00 mg.
M V I 1 2	10.00 ml.
F O L A T E	.00 mg.
K O N A K I O N	.00 mg.
T R A C E	.00 ml.
I O D I D E	.00 umol.
A L B U M E N	.00 ml.

```
G I V E N   O V E R   hrs.   V O L U M E  2000 ml.

        Enter [S] to save TPN order
        Enter [Y] to calculate TPN ingredients
```

Figure 18 : Order Requirements Screen

The order requirements information displayed on the screen may be amended by moving the cursor to a selected field and modifying the value. When the order is considered to be correct, a command "S" to "save" the order is entered, and the order is stored on the order file and printed for a permanent physical record.

The rule cluster associated with determining energy and protein requirements of the patient contains rule sets for each of the patient classifications. For the 'normal, starved' patient classification, there exists a rule to determine the mobility of the patient in order that adjustments to the energy and protein levels in the initial order may relate to the patient's needs. The level of mobility is taken from the assessment value in the patient screen data. In the current example, it was determined that the patient was bed-ridden and so the associated rule is :-

```
RULE : calculate_order::normal_patient_bed_ridden

            IF in correct context
            AND patient information exists for this mrn
            AND metabolic_status is NORMAL
            AND mobility is BED-RIDDEN

            THEN set protein and energy values
            AND set context to next
```

In OPS5, the rule appears as :-

```
    (p calculate_order::normal_patient_bed_ridden
         { <This_activity> (activity ^taskname base_order)
         { <This_patient> (patient_details ^mrn <mrn>
                                          ^metabolic_status NORM
                                          ^mobility BED) }
         { <This_requirement> (patient_requirement ^mrn <mrn> ) }
-->
    (modify <This_requirement> ^mrn <mrn>
                                          ^protein 0.7
                                          ^energy 23 )
    (modify <This_activity> ^taskname adjust_base_order))
```

Having set up the base protein and energy requirements, the control mechanism directs the firing of rules such as the following :-

RULE : calculate_order::adjust_protein

```
        IF in correct context
        AND patient information exists for this mrn
        AND protein and energy values exist
        AND order exists for this mrn and date

        THEN set order levels for protein ingredients
        AND set context to next
```

In OPS5, the rule appears as :-

```
    (p calculate_order::adjust_protein
         { <This_activity> (activity ^taskname adjust_base_order)
         { <This_patient> (patient_details ^mrn <mrn>
                                          ^weight <weight>
                                          ^complication << stress_low
                                            stress_moderate stress_severe >> ) }
         { <This_requirement> (patient_requirement ^mrn <mrn>
                                          ^protein <protein>
                                          ^energy <energy>
                                          ^fat <fat>
                                          ^drip_rate <drip> ) }
```

```
        { <This_order> (required_details ^mrn <mrn>
                                  ^order_date <required_date> ) }
-->
    (bind <required_amino> (compute <weight> * <protein> )
    (modify <This_order> ^amino <required_amino>
                                  ^dextrose (compute <required_amino> *
                                    <energy> )
                                  ^fat <fat>
                                  ^drip_rate <drip> )
    (modify <This_activity> ^taskname electrolytes) )
```

The final rule set used to calculate the base order for the medical staff consists of rules such as the following :-

```
RULE : calculate_order::electrolytes

            IF in correct context
            AND patient information exists for this mrn
            AND an order exists for this mrn and date

            THEN set electrolyte values
            AND set context to next
```

In OPS5, the rule appears as :-

```
    (p calculate_order::electrolytes
        { <This_activity> (activity ^taskname electrolytes)
        { <This_patient> (patient_details ^mrn <mrn>
                                  ^weight <weight>) }
        { <This_requirement> (patient_requirement ^mrn <mrn> ) }
        { <This_order> (required_details ^mrn <mrn>
                                  ^order_date <required_date> ) }
-->
    (modify <This_order> ^potassium (compute 2.5 * <weight> )
                                  ^sodium (compute 1.0 * <weight> )
                                  ^phosphate (compute 0.3 * <weight> )
                                  ^magnesium (compute 0.1 * <weight> )
                                  ^zinc (compute 0.02 * <weight> )
                                  ^calcium (compute 0.1 * <weight> )
                                  ^insulin (compute 1.0 * <weight> )
                                  ^vitamin_c 500
                                  ^thiamine 100 )
    (modify <This_activity> ^taskname display_base_order))
```

5.3 Pharmacy Calculation

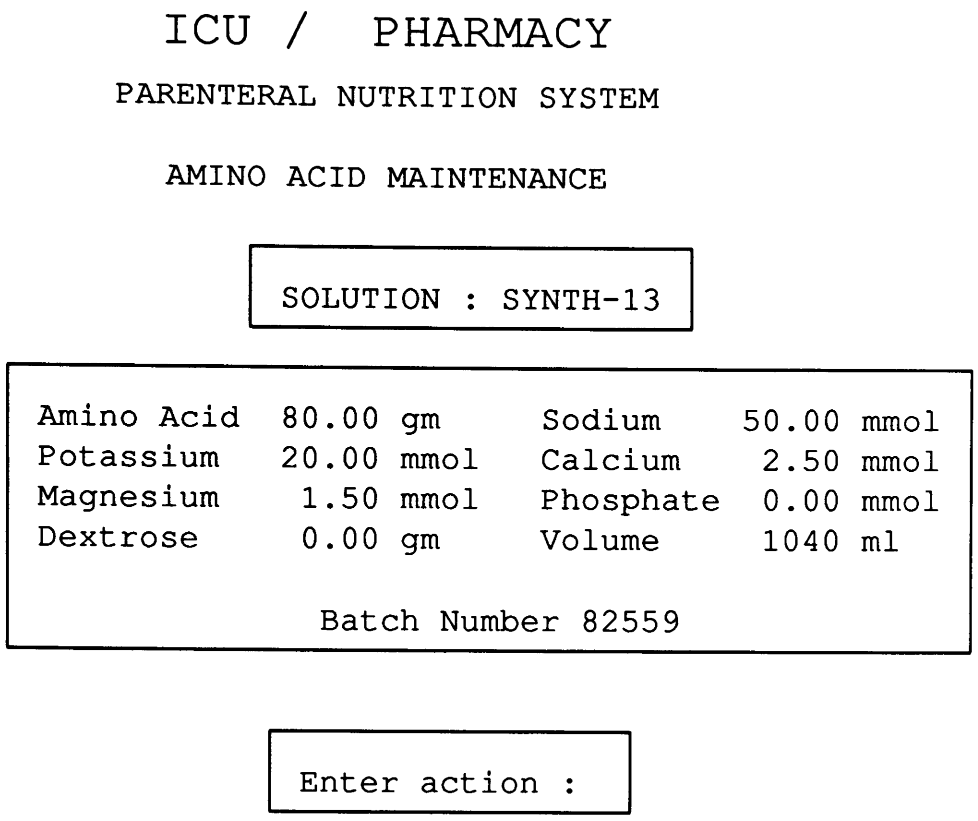
On entry to selection 2 of the main menu, the screen layout shown in Figure 15 is displayed. This screen form requests the user to enter the medical record number and date of the order to be produced. Only a single order may be produced within any 24 hour period and so the default date displayed is the current date.

After the medical record number and the date are entered, if a previously generated order exists, the details are displayed as shown in Figure 18. If an existing order does not exist, the system creates an empty order record on the order file and the pharmacist is required to enter the required order amounts into the relevant fields on the screen. The pharmacist then enters additional information such as infusion rate in the 'GIVEN OVER' field and the proposed solution volume into the next adjacent field, and then finally enters the command "Y" to calculate the ingredients. The resultant information is then stored on the order file and the TPN order printed onto an adhesive label to be placed onto the TPN mix container. The resulting TPN label is shown in Figure 20.

```
PARENTERAL NUTRITION ORDER                           23-Apr-90/ 02:55 PM
MRN    -   123456                                          Ward  -   ICU
Name   -  BLOGGS / FRED
Age    -  57      Sex    - M                          Weight - 100.00 Kgs
Batch Date - 230490        : ADDITIVE      UNITS    VOLUME        BATCH
-------------------------------------------------------------------------
Amino Acid    80.00 gm     : SYNTH-13        2.0    1040.0        52513
Dextrose     500.00 gm     : DEXTROSE50      0.0    1080.0        A1525
Fat            0.00 gm     : FAT - 10%       0.0       0.0
Phosphate     15.00 mmol   : KPO4            1.0      10.0
Sodium        35.00 mmol   : NACL            1.0      10.2
Potassium     90.00 mmol   : KCL             2.4      24.2
Magnesium      2.50 mmol   : MGSO4           0.3       1.3
Calcium        0.00 mmol   : CACL            0.0       0.0
Zinc           0.08 mmol   : ZNCL2           2.0       2.0
Insulin       80.00 Unit   : INSULIN         0.8       0.8
Thiamine       0.00 mg     : THIAMINE        0.0       0.0
Vitamin_C    500.00 mg     : VITAMIN_C       1.0       5.0
MVI12         10.00 ml     : MVI12           1.0      10.0
Folate        15.00 mg     : FOLATE          1.0       1.0
Konakion      10.00 mg     : KONAKION        1.0       1.0
Trace          0.00 ml     : TRACE           0.0       0.0
Iodide         0.00 uml    : IODIDE          0.0       0.0
Albumen        0.00 ml     : ALBUMEN         0.0       0.0

                           : WATER         None Req'd
-------------------------------------------------------
Volume : 2000 ml           : Total Volume         2185.5
Given Over 20 hours
Ordered by  - DJM
Prepared by - JJK                   Checked by _________
Time _____:_____ am/pm     : Date  23-04-90
```

Figure 19 : TPN Label Layout

The maintenance screens used to modify component levels of the ingredients used in manufacturing the TPN solution are shown in Figures 21 and 22. These modules were written in VAX BASIC.

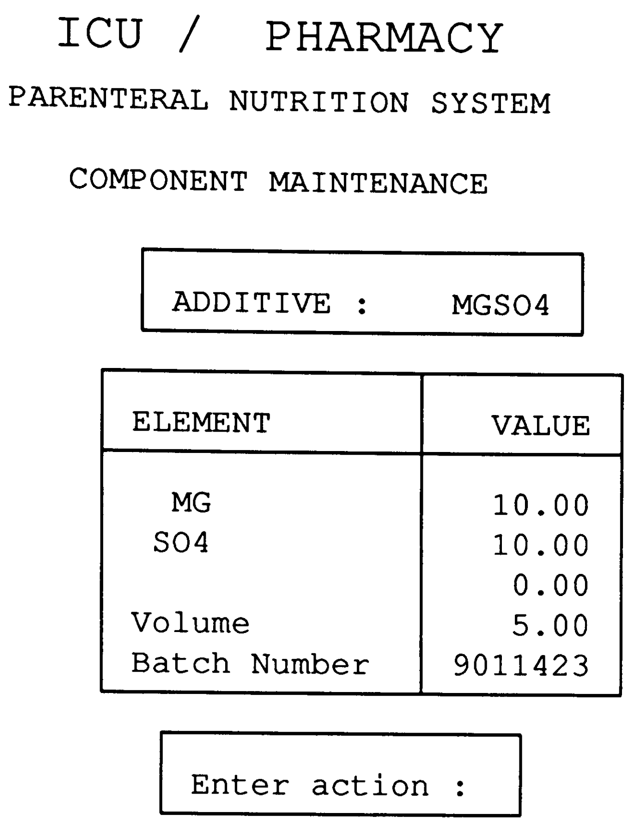


```
                 ICU  /  PHARMACY
           PARENTERAL NUTRITION SYSTEM
             AMINO ACID MAINTENANCE

              SOLUTION : SYNTH-13

   Amino Acid   80.00 gm     Sodium     50.00 mmol
   Potassium    20.00 mmol   Calcium     2.50 mmol
   Magnesium     1.50 mmol   Phosphate   0.00 mmol
   Dextrose      0.00 gm     Volume      1040 ml

               Batch Number 82559

               Enter action :
```

Figure 20 : Amino Acid Maintenance Screen

The amino acid and electrolyte component data used in the calculations are loaded into specific OPS5 amino and componentstructures via VAX BASIC subroutines. Figure 20 details the ingredients that are contained in 'SYNTH-13', one of the standard amino acid premix solutions.

From the order shown in Figure 18, 80 grammes of amino acid are required in the final TPN solution. The amino acid contains 80 grammes of amino acid together with specific amounts of additional elements and compounds in a total volume of 1040 millilitres. All these measures are subtracted from the order requirements values.



ICU / PHARMACY

PARENTERAL NUTRITION SYSTEM

COMPONENT MAINTENANCE

ADDITIVE : MGSO4

ELEMENT	VALUE
MG	10.00
SO4	10.00
	0.00
Volume	5.00
Batch Number	9011423

Enter action :

Figure 21 : Additive/Electrolyte Maintenance Screen

The next item calculated is the amount of dextrose. As with the amino requirement, units of 'Dextrose-50' are added to the calculation until sufficient dextrose is available to match the order. Since 'DEXTROSE-50' contains 250 grammes of the compound dextrose in a volume of 540 millilitres, 1080 millilitres of 'Dextrose-50' is required to be added to the solution.

Further down the order list is 'Magnesium'. The breakdown of the electrolyte additive magnesium sulphate is shown in Figure 21. Since the previous addition of the amino acid had reduced the 2.5 millimoles of elemental magnesium required, only 1.3 millimoles of magnesium sulphate was needed to be added.

The additives are processed in strict order from amino acid to albumin, adding the maximum allowable primary element or compound to the required values, at the same time reducing the lesser components in the order by the matching elements and compounds contained in the additive. The calculations proceed down the list of additives shown in Figure 19, until a result is achieved. Demon rules are used to watch over the calculations and determine when critical components are out of range.

The demon rule may be defined as :-

```
RULE : demon::negative_amino

            IF any electrolyte values are negative

            THEN print error message "USE ANOTHER AMINO ACID"
            AND remove order WME
            AND set context to start order again
```

In OPS5, the rule appears as :-

```
   (p demon::negative_amino
;
;  Calculate the required component levels from the amino acid
;   electrolytes. If not enough electrolytes are produced,
;   trap error and force recalculation.
;
{ <This_activity> (activity ^taskname calculating
            ^subtask amino) }
{ <This_order> (required_details ^mrn <selected_mrn>
           ^order_date <selected_date>
           ^phosphate < 0 ^sodium < 0
```

```
          ^calcium < 0 ^magnesium < 0
          ^potassium < 0) }
-->
    (write (crlf) |Too much of at least electrolyte -
          use another Amino Acid|)
    (remove <This_order>)
    (modify <This_activity> ^subtask amino))
```

CHAPTER 6

EVALUATION OF PERFORMANCE

6.1 Testing and Validation

Having discussed the concepts of the expert system in the previous chapters, it is now time to describe the system's performance in order to demonstrate that the developed system has achieved its intended goals.

6.1.1 Comparison with the Human Expert

The question can prompt a variety of answers, especially when looked at from the sometimes differing perspectives of the developer and the end-user. In part, this difference stems from the expectations that are initially set for the performance of the expert system.

The pharmacy expert sought absolute fidelity, and therefore found that an automated version of the person's expertise incomplete and lacking in common sense issues. The nature of the pharmacy problem domain was well defined, and consequently, the expert's perception of absolute success was correspondingly high. However, as a user, the pharmacy expert tended to judge the system differently. That is, the automated system was accepted as better than no system, even if it was perceived that a less than perfect result may be produced.

The medical expert was less demanding of the absolute result. The assistance offered by the expert system was appreciated where the situation, and consequently the result was straightforward. The nature of the medical expert system problem was less defined and consequently, the medical expert was prepared to accept an adequate result.

6.1.2 Specific Performance Criteria

There is always a danger of attempting to oversell the concepts of expert systems, and so it was considered more practical to take a benefits-oriented approach and focus on specific performance criteria such as :-

1) How useful the system is (in terms of productivity savings etc.)

2) How easily the system can be integrated into already existing (traditional) computing environments.

3) How the system can be made as user-friendly as possible and offer the presentation facilities that most users are familiar (menus, windows etc.).

With this perspective of performance testing and validation in mind, the following six specific criteria for testing and validating the expert system were considered :-

1) Accuracy
2) Completeness
3) Reliability and Consistency
4) Effective Reasoning
5) User-friendliness
6) Run-time Efficiency

6.1.2.1 Accuracy

Without question, this was considered to be the most important criteria in judging the expert system. However, it was considered that the accuracy of the result was slightly less important in the medical area (TPN order production) than in the pharmacy area (TPN formulation and manufacture), due to the mandatory checks that take place in the pharmacy.

One of the major advantages of a computer-based TPN system was that it guaranteed a complete, legible, consistent adhesive label on each container of TPN solution, rather than the traditional hand-written label. Prior to implementing any computer-based TPN system, a small survey was conducted in the pharmacy, in which twenty TPN solution labels were examined. Each of the handwritten labels was judged as to its completeness. A complete label was defined as one having the names, or a reasonable abbreviation or indication, of the drugs added, the amounts added, and an indication of the time and date of preparation together with sufficient detail to identify the medical and pharmacy staff responsible for the solution production. Of the total of twenty mixtures examined, four (20 percent) failed to meet these criteria on one or more issue.

The ability of the system to accurately and consistently perform the required calculations was the most important factor from a performance point of view. The necessary calculations are all simple arithmetic operations; however, when numerous individuals, subject to varied interruptions and disruptions, attempt to perform such calculations, the results are often less than acceptable.

6.1.2.2 Completeness

This criteria was related to the concept of accuracy, for as the expert system rule base grew in size and complexity, it tended to become more fallible under certain conditions. It was felt that most of this problem was related to the fact that a very high understanding of pharmacological practice was required together with a high degree of reasoning by both the expert and the knowledge engineer.

The difficulties arose since the knowledge engineer did not have a deep understanding of the pharmacy and medical aspects and the pharmacy and medical experts could not devote sufficient time to reason through the complex relationships that sometimes resulted in erroneous results. As mentioned previously in this chapter, it was accepted by both groups of experts that no expert system could ever really be complete, since it was accepted that the knowledge would be always changing and expanding.

6.1.2.3 Reliability and Consistency

Other than accuracy, the next most important criteria deemed important by the experts was reliability and consistency. Since this expert system was to be used as an 'assistant' and a teaching aide, it was considered that these two criteria would be in some situations (such as when an adequate result was acceptable) more important than accuracy.

6.1.2.4 Effective Reasoning

Normally, testing and validation focus primarily on system logic or the reasoning process, as this is the most straightforward area of the system to test. Unlike traditional programs, which depend on the programmer to control the logical processes within the system, an expert system relies on the structure of the rules and the user input to control the system.

In a rule-based system such as the one developed for the project, rule validation was performed by running sample or test cases that represented hypothetical but likely situations through the expert system. Using the debugging features of OPS5, a report of the rules that fired and a trace of what subgoals were reached was produced and checked by an expert to determine if such an expert would have solved the problem the same way, given the same data and circumstances. In particular, what was sought was the validity of the result and the order of the rules that fired. That way, not only was the accuracy of the system able to be tested and improved, but the expert system efficiency could also be assessed.

6.1.2.5 User-friendliness

As previously suggested, the user-friendliness of an expert system is often a key factor in its success. No matter how accurate, complete, reliable and consistent the system may seem from the developer's perspective, it may still be of little use to the user if it cannot convey its knowledge or expertise effectively or if it is difficult to use and fails to gain sufficient user acceptance.

Throughout the design of the expert system, an attempt was made to reduce the amount of input required from the user and rely on predefined values where possible. This feature was practical within the pharmacy module since this department was keen to follow defined standards of practice thereby reducing the possibility of incorrect results.

Due to the nature of the problem in the medical module, a certain level of data entry was required, which in fact lead to a very low level of user acceptance. On-line access to patient laboratory results and the complete patient demography would have reduced a large part of this data entry to an acceptable level.

In general, most users found the system useable, but at the same time suggested that they would prefer to use features normally seen in PC-based expert system shells, such as detailed on-line help, explanation reasoning and more visually stimulating screen design (windows, shadowed pull-down menus, colour highlighting).

6.1.2.6 Run-time Efficiency

Once the more important elements of the expert system had been tested and validated, the run-time efficiency was considered.
The performance of the system was not considered to be a problem to the expert system running on a mid-range minicomputer, however, some time was devoted to studying the search strategy.

As one would expect within the recognize-act cycle of match, conflict resolution and act, the match phase accounts for up to 90% of the execution time. This fact is hardly surprising since every working memory element is compared to every condition element in every rule. For example, given a medium-sized OPS5 program with 100 working memory elements and 100 rules each containing five condition elements would require 50,000 comparisons, using the 'brute-force' method. This would obviously account for much of the execution time. Since OPS5 is based on the efficient match algorithm called the RETE match algorithm, a basic understanding of the algorithm is essential to an understanding of efficiency in OPS5.

CHAPTER 7

CONCLUSIONS AND FUTURE

7.1 Summary and Conclusions

An experiment in the application of expert system techniques to the area of patient treatment within a hospital has been described. The problem chosen was the task of preparing a detailed prescription used to prepare intravenous solutions by a hospital pharmacist and the implementation was carried out in OPS5 and BASIC. This work was carried out as the project component of the thesis and as such it was never intended that the system would be used "live" by medical or pharmacy staff within the Illawarra Area Health Service. The primary motivation for the project was to gain some experience with expert system techniques and with the OPS5 language. Although the project is finished and the system is never likely to be developed further by the Illawarra Area Health Service, it has shown what is possible with an approach that had previously never been considered.

Although planned partly to gain some experience with OPS5, the project has made extensive use of an algorithmic language, BASIC. The principle reasons for this were the need to search databases efficiently and the limited arithmetic facilities of OPS5. It was also necessary to use BASIC for access to screen management and summary reporting facilities.

In retrospect, it was agreed that developing one module of the system rather than both the medical and pharmacy modules would have produced a substantial project suitable for research. As has been reported previously, the pharmacy calculation module was developed to completion, however, the complexity of the medical module restricted its depth of expertise and level of development.

7.2 The OPS5 Programming Language

It was found that OPS5 had a limited but simple syntax that was easy to learn. The difficulty in using OPS5 was found to lie in becoming accustomed to how the rule interpreter functioned and in learning how to write rules that produce meaningful results. Despite this complexity, OPS5 was found to be a good prototyping tool since the programmer can concentrate on understanding and representing the domain knowledge and less on control statements.

The language OPS5 was found to be convenient for some aspects of the implementation, particularly the encoding of the expertise of the medical staff. For example, it was found easy to add further expertise incrementally as testing forced further discussions with medical and pharmacy experts to refine the knowledge contained in the expert system. The OPS5 system was sufficiently efficient to operate as a practical application.

The interface between OPS5 and BASIC (or the VAX operating system in general) proved very difficult to master at first and very tedious once mastered. Furthermore, the kind of side-effects that the BASIC modules must have on the working memory of the OPS5 modules is error-prone and makes the entire expert system difficult to understand.

The lack of standard mechanisms for using certainty factors to express relative degrees of belief among hypotheses and the lack of ability to explain the flow of reasoning to the user was deemed to be a major deficiency of the language. These facilities were required to be developed from scratch leading to unnecessary development effort to the project.

It was shown that the system performed well when compared to the behaviour of experienced medical and pharmacy staff. Thus, it is concluded that such an expert system could be successfully employed as an advisor in medical related areas. Daily repetitive tasks could be delegated to less-skilled staff, releasing the highly-skilled to diagnose and manage the more complicated situations.

OPS5 highlighted some notable differences to most other computer languages that are not rule-based. The flow of control in OPS5 is not expressed in explicit control statements. The language is data-driven; the rule interpreter chooses the rule to execute depending on the data that match the rules. There are no conditional statements, calls to OPS5 subroutines or procedures, nor any iterative loops. When the program executes, rather than there being a single thread of control that solves the problem, there can be many parallel control paths. It is not useful to trace the paths through the program by looking at the code alone.

The entire state of an OPS5 program is described by the contents of a global working memory area which results in all rules are matched against all data. This means that a program cannot refer to some specific data which relates to a subset of rules and so ignore specific data. The differences in control structures and program state required a different approach to be taken during the development of the OPS5 programs compared to more familiar programming techniques used for sequential languages.

7.3 Expert Systems in a Medical Environment

The two types of information that medical staff need to make medical-related decisions are data and knowledge. Examples of knowledge include the medical textbook information and heuristic knowledge acquired from physicians. Most knowledge-based systems concentrate on the knowledge half of the problem - how to gather it, represent it and reason from it. This type of background information is necessary, but not sufficient for doctors to make medical decisions. Diagnosis and treatment of patients also require information about the patients themselves.

Medical knowledge, in general, is low-density knowledge. That is, it is necessary to tell a knowledge system a great deal of information to characterise a patient so the knowledge system can do useful work.

There are many medical problems that do not require large amounts of data for their resolution. These cases, however, are not generally difficult. The difficulty of the diagnosis tends to be directly related to the amount of information required to make a diagnosis, and so the only practical knowledge system is one that has automatic access to a patient database, where the knowledge system is able to search for all available information about a patient rather than request missing information from the physician. The usefulness of any medical knowledge system is limited if each consultation may require ten minutes of data-entry and dialogue to complete a diagnosis.

During the latter stages of the project, an unexpected advantage to medical knowledge-based systems was encountered. The medical staff using the system agreed that the knowledge systems tended to take the drudgery out of medicine. It seems that some of the physician's jobs are pretty boring, with many tasks considered simple and repetitive. Interpreting most test results were considered to be straightforward. With a knowledge-based medical consultant around, the physicians felt that they would have the challenge of looking at the more interesting cases and could concentrate on the more difficult ones.

7.4 Future Research

Many of the early, pioneering examples of expert systems such as DENDRAL [BUCHANAN79], MYCIN [SHORTLIFFE76], R1 [MCDERMOTT80] and PROSPECTOR [GASCHNIG79] were mostly products of universities and research laboratories and were developed to solve problems in domains which were substantially removed from the main business of commercial information processing. The effect of these widely reported expert systems has been to tempt data processing professionals to consider analogous applications in their own area for possible expert systems development. This atmosphere of cautious optimism has been summed up by J.K. Debenham [DEBENHAM89], who wrote :-

"It is fair to comment that this search for analogous application areas in main stream computing has not always been successful, and this failure to identify worthwhile, analogous problem areas has led to the general conclusion that expert systems are still 'some way down the track'".

Expert systems will be included as part of the systems developed in the future. Those organisations with the foresight to realise this and plan will gain a very competitive advantage. This is shown, in part, by the secrecy which currently shrouds expert systems development in the insurance and finance industries and the appearance of data-base management systems which contain integrated expert system shells.

The perception of what an expert system is, and what it does still seems to be clouded in some sort of mystique, at least in Australia. The slow acceptance of expert system technology seems to be due to the fact that many aspects of expert systems contradict traditional computing beliefs.

An expert system gives an "adequate" solution to a problem as opposed to the traditional "absolute" solution required in many applications. The expert system often has to be prototyped and then modified as new knowledge comes to hand and the system may never be completed. Knowledge engineering and expert systems should be considered to be just another piece of software, to be maintained just like any other system.

GLOSSARY

Backward Chaining	A type of system activity that attempts to solve a problem by stating a goal and looking into the database for the conditions that would cause that goal to come about, then reiterating this process, using those conditions as the goals and searching for their preconditions.
Central Venous	Via the main blood vessel entering the heart.
Domain	The problem area about which a system has knowledge.
Electrolyte	The ionic component of bodily fluids (eg. K+, HPO3-).
Enteral	Passed into the stomach via mouth or nose.
Explanation Facility	A feature of many expert systems that tells what steps were involved in the process by which the system arrived at a solution.
Forward Chaining	A type of system that applies operators to a current state in order to produce a new state, and so on, until a solution is reached. In an expert system, a forward-chaining rule detects certain facts in the database and takes an action because of them.
Frame	A knowledge representation technique based on the idea of a frame of reference. A frame carries with it a set of slots which can represent objects that are normally associated with the subject of the frame. The slots can then point to other slots or frames, a feature that gives frame-based systems the ability to allow one object to inherit characteristics from another, and to support inferences.
Heuristic	A process, sometimes a rule of thumb, that may help in the solution of a problem, but that does

	not guarantee the best solution, or indeed, any solution. Because the success of a heuristic is not guaranteed, a problem that can be solved by one algorithm frequently requires many heuristics. The primary effect of heuristics is to eliminate the need to examine every possible approach.
Hyperalimentation	Excess nutrition (less used term for TPN).
Inference	A conclusion based on a premise.
Inference Engine	The part of a rule-based system that selects and executes rules. In contrast to algorithms embedded in traditional software programs, but like the human reasoning process, the conclusion that an inference engine will draw from a given set of facts is not known in advance.
Intravenous	Into a vein or veins.
Knowledge Base	The part of an artificial intelligence system that contains structured, codified knowledge and heuristics used to solve problems. Artificial intelligence systems using such a base are called knowledge-based systems. In an expert system, the knowledge base generally contains a model of the problem, knowledge about the behavior and interactions of objects in the problem domain, and a level of general-purpose knowledge.
Knowledge Engineer	A person who implements an expert system.
Knowledge Representation	A structure in which knowledge can be stored in a way that allows the system to understand the relationships among pieces of knowledge and to manipulate those relationships.
LISP	A programming language (LISt Processing) designed specifically to manipulate symbols rather than numeric data.
Nasogastric	Passage of substances via nose or mouth.

Parenteral Nutrition	Nutritional regimes which provide more nutritients than the usual dextrose and saline solutions, but not a high energy balanced diet.
Pattern Matching	A process performed by an expert system during its search through its knowledge base.
Physiology	The science of the vital phenomena and organic functions of animals and plants.
Production Rule	A procedural process triggered by a pattern. Rules are commonly structured in an if...then...format. If the pattern is matched, then schedule a procedure for execution.
PROLOG	A programming language (PRO-gramming in LOG-ic) designed primarily to manipulate symbols rather than numeric data. PROLOG differs from LISP in its approach. PROLOG programs use assertions about objects and relationships to handle queries about them and anwers these queries by consulting its knowledge base of relations.
Total Parenteral Nutrition	TPN - Administration of the components of a normal diet intravenously, including water, amino acids, carbohydrate, fat, vitamins, minerals and trace elements. In adults the administration of 2,000-3,000 kcal (non-protein) per day is implied.
Working memory	The dynamic portion of a production system's memory. Working memory contains the database of the system, which changes as the rules are executed.

REFERENCES

[BROWNSTON85]	Brownston,L., Farrell,R., Kant,E. and Martin,N., "Programming Expert Systems in OPS5", Addison-Wesley Publishing Company, Reading, Mass., 1985.
[BUCHANAN79]	Buchanan, B.G. and Feigenbaum, E.A., "DENDRAL and Meta-DENDRAL : their applications dimension", Journal of Artificial Intelligence", pp5-24.
[BUCHANAN84]	Buchanan, B. G. and Shortliffe, E. H., "Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project", Reading, Massachusetts, Addison-Wesley Publishing Company, 1984.
[CRI86]	Smart,G. and Langeland-Knudsen, J., "The CRI Directory of Expert Systems", Learned Information (Europe) Ltd., 1986.
[DAVIS77]	Davis, R., Buchanan, B.G. and Shortliffe, E.H., "Production Rules as a Representation for a Knowledge-Based Consultation Program", Artificial Intelligence, 8 (1), 1977, pp15-45.
[DEBENHAM89]	Debenham, J.K., "Knowledge Systems Design", Prentice-Hall, Sydney, 1989.
[DUDA80]	Duda, R. Gaschnig, J. Hart, P. "Model Design in the PROSPECTOR Consultant System for Mineral Exploration", in Expert Systems in the Micro-Electronic Age (D. Michie ed.), pp153-167, Edinburgh University, Edinburgh, 1980.
[FEIGENBAUM78]	Feigenbaum, E.A., "State of the Art Report on Machine Intelligence", A. Bond (Ed.), Maidenhead: Pergamon-Infotech, 1978.

[FORGY82]	Forgy, C.L. "Rete: A fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", Artificial Intelligence, 19 (1982) 17-37.
[GASCHNIG76]	Gaschnig, J., "Preliminary Performance Analysis of the Prospector Consultant System for Mineral Exploration", In Proceedings of the Sixth International Joint Conference on Artificial Intelligence, Tokyo, 1979, pp308-310.
[HAYES-ROTH83]	Hayes-Roth, F., Watermann, D.A. and Lenat, D.B., (eds.) "Building Expert Systems", Addison-Wesley Publishing Company, Reading, Massachusetts, 1983.
[KOWALSKI79]	Kowalski,R., "Logic for Problem Solving", North Holland, New York, 1979.
[KUMARA86]	Kumara, S. Soyster, A.L. and Kashyap, A.L. "An Introduction to Artificial Intelligence", Industrial Engineering, Vol 18, No 12, 1986, pp9-20.
[MCCARTHY83]	McCarthy, J. "Artificial Intelligence (AI)" Colliers Encyclopedia Vol 1, 1983, ed.: pp714-716.
[MCDERMOTT80]	McDermott,J., "R1:an expert in the Computer Systems domains", AAAI-80, pp269-271.
[MCDERMOTT82]	McDermott, J., "R1 : A Rule-Based Configurer of Computer Systems", Artificial Intelligence, 19 (1982), pp39-88.
[MCDONALD83]	McDonald,C.J. et.al. "The Regenstrief Clinical Laboratory System", IEEE Proceedings of the Seventh Annual Symposium on Computer Applications in Medical Care, 1983,pp. 254-257.
[NEWELL72]	Newell,A. and Simon,H., "Human Problem-Solving", Englewood Cliffs, NJ: Prentice-Hall, 1972.

[RAUCH-HINDIN86]	Rauch-Hindin, W.B. "Artificial Intelligence in Business, Science and Industry Volume I: Fundamentals and Volume II: Applications", Prentice-Hall, New Jersey, 1986.
[SHORTLIFFE76]	Shortliffe, E.H., "Computer-Based Medical Consultations : MYCIN", Elsevier, New York, 1976.
[WATERMANN86]	Watermann, D.A., "A guide to Expert Systems", Addison-Wesley Publishing Company, Reading Massachusetts, 1986.
[WEBSTER76]	Webster's Third New International Dictionary, G. & C. Merrian Company Publishers, Springfield, Massachusetts, 1976.
[WILLIAMS89]	Williams,G.J., "FrameUP: A frames formalism for expert systems", Aust. Comp. Journal, Vol. 21, No.1, February 1989.
[WINSTON81]	Winston,P. and Horn,B., "LISP", Addison-Wesley, Reading, Mass., 1981.