THE P.R.O. EXPERT SYSTEM SHELL

Thesis submitted by

JOHN BRADSHAW

In Fulfillment of the
Requirements for the degree

MASTER OF SCIENCE

Rhodes University
December 1986

Abstract

This thesis reports the research which led to the development of the P.R.O. Expert System Shell. The P.R.O. System is primarily, though not exclusively, designed for use in ecological domains. In the light of two specific expert systems, The RCS (River Conservation Status) and the Aquaculture Systems, which were developed as part of this research, a number of areas of importance have been identified. The most significant of these is the need to handle uncertainty effectively. The style of knowledge representation to be implemented also plays an important role. After consulting the relevant literature and the available microcomputer expert system shells, a number of ideas have been included in the P.R.O. System.

The P.R.O. System is a backward chaining, production system based expert system shell. It embodies a simple but effective method of handling uncertainty. An important feature of this method is that it takes cognizance of the different relative importances of the conditions which need to be satisfied before a conclusion can be reached. The knowledge base consists of more than rules and questions. It also contains meta-knowledge, which is used by the inference engine.

The P.R.O. System has been designed to be of practical use. Its strongest recommendations are therefore, that the two non-trivial systems which have been implemented in it, have been accepted by the experts and their peers as systems which produce good, accurate answers.

Acknowledgements

There are many people who have contributed to the success of this project. I am indebted to the following:

Denis Riordan - for all his support throughout the first two years of my academic life and particularly for the patience he showed whilst reading the initial drafts of this thesis.

Karen Wrench - who not only fed me during the year, but also gallantly accepted the task of proof reading this thesis.

Jay O'Keffee and Mike Bruton - without their initiation of the RCS and Aquaculture projects respectively, and without their general enthusiasm for this field of research, many of the ideas embodied in the P.R.O. System may never have seen the light of day.

Dale Danilewitz and Nick Dean - for their invaluable assistance in collating the knowledge for the RCS and Aquaculture systems respectively.

Pat Terry - for his good sense in taking long leave just when I needed a job, and for his fine choice of leave replacement!

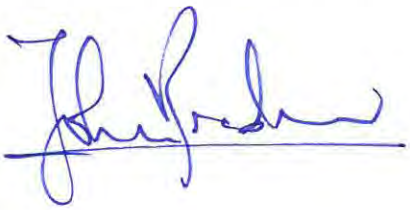
Molly Brooks - for all the helpful, little things she has done during the past two years.

To all, I offer my sincerest thanks and wish you a happy and peaceful Christmas.

JohnB.

Declaration

I hereby declare that all the work submitted in this thesis is a reflection of my own original research, except where it is clearly indicated to the contrary.



John Bradshaw

Monday, 15th December, 1986

Contents

1. Introduction

At the start of this project there were a number of ecologists interested in producing expert systems. The need for a general expert system shell for use in ecological applications was therefore clearly identified.

Several commercially available microcomputer expert system shells were initially considered by the ecologists. These were found to lack the flexibility which was required. As a result, the P.R.O. Expert System Shell (the P.R.O. System) was developed. The research for this project involved the investigation of the major areas of expert system design.

Considering the nature of ecological work, it was decided from the outset to target the system at the microcomputer environment. An efficient system was thus required. Early investigation also showed that ecological problems are characterized by the uncertainty of the raw data. The expert system shell would be required to handle this effectively. Thus a major emphasis of the research was devoted to the investigation of the current techniques and the development of an efficient and effective algorithm to handle this problem.

The P.R.O. System is a rulebased, backward chaining expert system shell. It provides an inference mechanism which is easy for experts, outside the field of mathematics and computer science, to understand. Although the P.R.O. System has been designed for use on ecological applications, it is not limited to this area at all.

The construction and evaluation of expert systems is difficult as many of the issues are subjective. However, ecologists who have used the P.R.O. System are satisfied with the results which their systems produce.

In order to achieve the desired degree of flexibility, as many of the

variables as possible have been placed in the knowledge base. This provides the expert with a framework which can be readily tailored to suit his individual requirements. The mechanism of the shell has been kept as simple and as transparent as possible. This reduces the conceptual problems which users may have with expert systems.

Associated with the P.R.O. Expert System Shell is the P.R.O. Knowledge Base Parser. The Parser detects syntactic and, to a limited extent, semantic errors in the knowledge base.

An important aspect of this research has been the handling of uncertainty. A method of propagating uncertainty values through an hierarchical set of rules has been developed and implemented. This method allows conditions to be inter-related, which is important when considering ecological applications where environmental factors are all linked. For the sake of efficiency, this method also attempts to overcome some of the computational expense problems which are associated with other methods.

In the ecological environment many of the figures which other statistical models require are simply unavailable. It is the more subjective knowledge which the ecologist often uses in his deliberations. This system attempts to capture and use this knowledge. Starfield et al [Sta84] recognize this problem: "On the one hand, the numerical models required information about the process and rates of change that the biologists could not provide; on the other hand, the biologists were in fact reacting to field observations which were very different from the substance of the models being built".

As part of this project, several systems have been implemented using the P.R.O. Expert System Shell. The first of these determines the conservation status of South African rivers [O'K86a]. This has been accepted by the relevant experts in this country as a system which produces good accurate answers. It is currently being used in an detailed survey of the water resources of the Eastern Transvaal

[NRI86]. A second system recommends a suitable species for aquaculture [Bru86a]. This system is still under development, but already the feedback has been most encouraging. The supervision of the development of these two systems was an integral part of the P.R.O. System project. Without them many of the ideas discussed in this thesis may never have seen the light of day. They also provided non-trivial examples on which to evaluate the ideas produced. The development of these two systems, therefore, constitutes an important part of the entire research reflected in this thesis.

An expert system to be used in shark taxonomy [For86] has been separately developed. It makes use of the same method for handling uncertainty as the P.R.O. System and, at this stage, produces results which correlate closely to those of the resident expert.

This thesis reports on all the issues discussed above and also looks at extensions and further developments which may result from the project. Throughout the thesis, reference is made to well known expert systems such as MYCIN [Buc84] and Prospector [Dud79], as well as to certain microcomputer expert system shells, notably Micro Expert [Cox82], Expert-Ease [Mic84] and Exsys [Hum84].

The research undertaken is documented as follows:

Chapter 2 introduces the concept of expert systems and expert system shells. The different physical characteristics of these systems are discussed. The issues of knowledge representation techniques are considered, with reference to semantic nets, frame based systems and production systems. Some of the problems which are involved in the design of expert systems are also identified.

The third chapter discusses the structure of the P.R.O. Expert System Shell knowledge base. A number of different knowledge base structures are available to the knowledge engineer (the person who 'codes' the knowledge base). Besides rules' and 'questions' (which form the backbone of the knowledge base). 'systemtype', 'hypothesis' and

weight knowledge base structures are also used. The syntax and semantics of these knowledge base structures are discussed in detail.

The fourth chapter looks at the structure of the shell components of the P.R.O. System. It is in this chapter that some of the technicalities of the system are discussed. The suitability of Prolog as a language for producing expert systems and expert system shells is considered. Furthermore, the differences between languages, expert system tools and expert system shells are highlighted. It is shown that expert system tools provide the knowledge engineer with the flexibility of languages as well as the structure of expert system shells. It is also shown that the use of Prolog facilitates the provision of this additional flexibility in expert system shells.

The problems associated with the handling of uncertainty are discussed in some detail in chapter 5. Several methods which have been implemented in existing systems are described and some of the problems associated with these different methods are highlighted. The chapter also introduces the mechanism which has been developed and implemented in the P.R.O. Expert System Shell. This chapter concludes by evaluating and comparing the method used in the P.R.O. System to the more generally accepted Bayes' and Dempster-Shafer methods of handling uncertainty. The problems associated with each of these two established methods are explained. The Bayes' theorem makes the assumption that there is no relationship between the conditions [Whi85]. The Dempster-Shafer algorithm suffers from the problems of computational intractability [Gor85]. The P.R.O. System mechanism for handling uncertainty makes no assumptions about the nature of the conditions or their inter-relationships. The relative importance parameter is used to express the relative weighting to be assigned to each condition in a rule.

Chapter 6 deals with the P.R.O. Knowledge Base Parser. This is a system which has been developed to aid the knowledge engineer in setting up his knowledge base. The Parser has been developed to reduce the harshness of the knowledge acquisition task. The interface

between the knowledge engineer and the P.R.O. System is somewhat crude and there is definitely scope for more work in this area. The chapter explains that the restrictions in the use of clauses in Turbo Prolog were chiefly responsible for this crudity.

Chapter 7 describes two expert systems implemented using the P.R.O. System. These are the RCS (River Conservation Status) System [O'K86a and b] and the Aquaculture System [Bru86a]. The RCS System is designed to aid conservationists in the complex task of determining the conservation potential of a South African river. It has been developed over the period of approximately one man year and consists of 200 rules and questions. The Aquaculture system is still under development. To date. the system is able to advise on the suitability of fourteen different fish species, given the physical conditions of a particular site and the amount of finance available. Both systems are referred to throughout the thesis, since their development constituted an integral part of the whole project.

The eighth and final chapter consists of a resume of the important features of the P.R.O. Expert System Shell. This chapter explains that although the P.R.O. System's method of handling uncertainty is not derived mathematically. it has been shown to work in three non-trivial systems. The P.R.O. System attempts to reflect the subjective quality exhibited by the experts. It is asserted that this is perhaps the reason why the mechanism has performed well in real life situations.

Appendix A contains a detailed description of a small example tax system knowledge base. This system is used to illustrate certain concepts dealt with in the thesis. It is suggested that the reader should familiarize himself with the example tax system and both the RCS (River Conservation Status) and Aquaculture Systems (discussed in chapter 7) before reading the main sections of this thesis.

Appendices B and C contain tracings of the P.R.O. Expert System Shell functions and of the P.R.O. Knowledge Base Parser respectively.

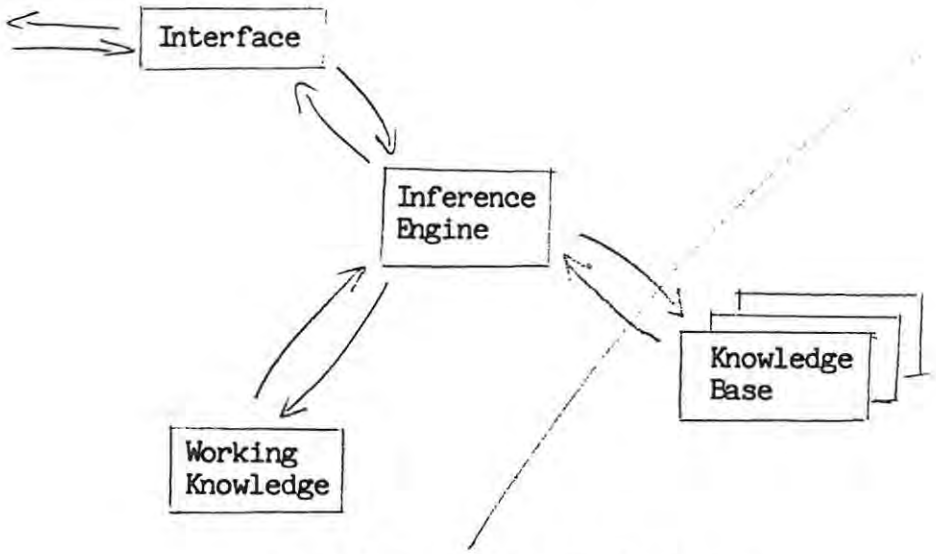
2. A Brief Overview of Expert Systems

This chapter provides an introduction to expert systems and expert system shells. It discusses the issues of knowledge representation and knowledge acquisition. The aim is to place the work described in the rest of this thesis in perspective. A number of texts [Buc84, Cha85, Dav75, Hay83, Jac86, Wei85] provide introductory material in this area of artificial intelligence research. They describe fully the issues involved and the different construction techniques which have been developed. Several journals provide comprehensive information on the state of expert system research. These include the Journal of Artificial Intelligence and the AI magazine, which publish papers of a more technical nature. Conferences, such as those organised by the British Computer Societies Special Interest Group on Expert Systems, are devoted to the advancement of expert systems. The resulting proceedings [ES83, Bra85a, Mer85] discuss current aspects of expert system research in depth.

Expert systems are computer based systems designed to emulate the problem solving action of domain experts. Early research attempted to produce intelligent systems which could solve a wide range of problems. Newell and Simon's GPS system is an example of these attempts [Win77]. More recently the trend has been to develop more restricted, domain specific systems. Humans are often experts only in a narrow domain and the trend in expert system design has followed the same restrictions [Wat86].

Although the chief objective in expert system design is to develop systems which produce the same results and problem solving performance as human experts, the internal workings of these systems may differ radically from the processes that humans use. Some of the most successful systems do, however, use production systems [Dav75] which embody a natural approach to problem solving. This chapter describes the structure of expert systems and the different forms of knowledge representation.

An expert system is composed of four components. These are the knowledge base. the inference engine. the working knowledge and the user interface. Figure 2.1. illustrates the relationship between these components. Each component is discussed in detail below.



Typical Expert System Structure

This figure illustrates the relationship between the four components of an expert system. The inference engine is the controlling system. It invokes the interface section when user input is required. The inference engine works through the knowledge base; its operations are determined by the knowledge base and the answers it receives to questions asked. Intermediate results are stored in the working knowledge section.

Fig. 2.1.

A recent trend in expert system development has been to produce expert system shells and tools [Jac84]. Expert system shells are systems consisting only of the user interface. the inference engine and the working knowledge components. The rationale behind this approach is that a single shell system should manage a wide range of knowledge bases. Expert systems are expensive software products and the use of an expert system shell helps minimize the cost of developing specific expert systems.

2.1. The Knowledge Base

The knowledge base component represents the domain knowledge of the expert explicitly. but may contain more than the raw knowledge of the expert. It may also contain other. more general forms of knowledge. such as meta-knowledge (knowledge about knowledge) or self knowledge [Wat86]. which the inference engine uses to explain the deductions of the expert system at runtime.

The structure of the knowledge base must be in a form which the inference engine can use. The implementation of a particular knowledge base structure conforms to the classic chicken and egg situation. On the one hand. the structure will effect the design of the inference engine and conversely. the inference engine will determine the structure of the knowledge base. A compromise needs to be made between the two. The inference engine must be able to work with the knowledge base and the knowledge base must be able to hold the knowledge it needs.

Knowledge. often regarded as purely declarative. can also be conceptualized as being procedural. Declarative knowledge is 'knowing what (for example the colour of the bird) and procedural knowledge is knowing how (for example knowing that larks are predominantly brown).

Users of expert systems often assume that the problem solving methodology displayed by an expert system is inherent in the structure and design of the inference engine and thus the product of the inference engine designer. The relationship between the inference engine and the knowledge base is thus misunderstood. The methodology which is perceived is actually described in the knowledge base. This is a reflection of the problem solving approach used by the domain expert. The knowledge engineer defines the knowledge base and so it is he who defines the problem solving methodology to be used by the system.

The relationship between a car and its driver is analogous to the relationship between the inference engine and the knowledge base. The designers and manufacturers of a car produce a vehicle with a set of standard features. The driver determines precisely where the car will go and how it will get there. For example, the design of a car does not determine a person's route to work; this is a function of the driver. The inference engine (the car) provides certain basic features. Within the restrictions of these features the precise operation of the inference engine is determined by the knowledge base (the driver).

The challenge the knowledge engineer faces is, within the framework determined by the inference engine, to reproduce the problem solving methodology of the human expert in the knowledge base. The more accurately the expert system portrays the problem solving methodology of the expert, the better the system.

The knowledge base structure should therefore enable the knowledge engineer to represent the problem in a natural way. It is important that the inference engine gives the knowledge engineer the flexibility to represent effectively the knowledge of the domain expert. To continue the above analogy, racing cars need powerful engines and sleek, light bodies to enable the driver to achieve his set objectives. In an analogous way it is important that the inference engine be designed in such a way that the knowledge base structure is suitable to accommodate the knowledge of the domain expert. One of the major issues in expert system design is, therefore, the choice of knowledge base structure.

2.1.1. Representation of knowledge

A number of different knowledge representation techniques are discussed widely in the literature. The most important of these are semantic networks [Wat86], frames [Aik83] and production systems [Dav75]. These techniques are described in the following sections.

2.1.1.1. Semantic nets

Donald Waterman defines a semantic net as "a knowledge representation method consisting of a network of nodes standing for concepts or objects connected by arcs (links) describing the relations between the nodes" [Wat86]. The important aspects of this representation are that particular relationships between facts (objects or concepts) are directly represented by explicitly linking the two facts together. The responsibility for the interpretation of the links lies with the inference system used [Bar81].

Semantic nets therefore contain information about the relationships between facts and not explicit information about the facts themselves. Recent work in this field, for example the Centaur system [Aik83], incorporates the use of frames (described below) to replace nodes.

2.1.1.2. Frames

This style of knowledge representation was originated by Marvin Minsky. He describes a frame as "a data structure for representing a stereotype situation. Attached to each frame is information about how to use the frame, about what to do next and about what to do if the expectations in the frame are not confirmed" [Min75].

Each frame contains a collection of related information. The information is contained in two parts, namely the attribute name and its value part (called a slot). As attribute values (such as colour) are found, so the corresponding slots are instantiated. As slots are filled, emptied or interrogated, procedures linked to the slots are initiated.

A frame may be concisely described as "a data structure which contains both procedural and declarative information in predefined internal relations" [Bar81]. The adjective declarative implies that the frame contains information about the object (for example its colour), while procedural implies that the frame contains information about

what to do - as described above.

2.1.1.3. Production systems

The basic unit of knowledge used in a production system is a rule [Dav75. Wat76]. A rule takes the form

```
If A and B
     then C
```

This can be read as "if conditions A and B are true then C must also be true". The if-parts (A and B) of the rule trigger either other rules or questions. When the rule is invoked the system will try to satisfy all the conditions (the if-parts). The rule is not affected by the processes which evaluate these conditions; it is affected only by the results. The then part of the rule (C) may represent certain actions to be taken if all the conditions are satisfied.

The advantages which accrue from the use of production systems stem from the ease with which rules can be interpreted by a human. The relative simplicity with which the knowledge engineer can 'code' his knowledge base in the first instance, and thereafter. the ease with which it can be modified can be attributed to the natural approach of production systems.

A number of successful systems have been developed using production systems. Foremost amongst these are MYCIN [Buc84]. and Prospector [Dud79]. MYCIN is a system. developed at Stanford University, which gives diagnostic advice to patients suffering from infectious diseases. Prospector is a system which gives a prognosis on the favorableness of an exploration site for the occurrence of significant ore bodies of a given type.

The advantages of using production systems is that rules embody a natural style of problem solving. Therefore. the expert and the user. who are often computer illiterate. can conceptualize the inner workings of the expert system more easily. Current research work at

the University of Cape Town has investigated semantic nets, production systems and frame based systems. The results, as yet unpublished, indicate that Frame based systems seem to offer the most benefits. Work has begun on developing a frame based system in common Lisp for a microcomputer [Jco86].

2.1.2. Knowledge acquisition

An important phase in the design of an expert system is the collection of the expert knowledge [Gre83]. This knowledge includes the expert s years of experience, in addition to his formally' acquired knowledge. The knowledge must also be collated in a form which will combine most effectively with the knowledge base structure.

There are two sides to the problem of knowledge acquisition. In the first instance, many experts find it difficult to articulate their expertise. This is particularly a problem if the knowledge engineer is unfamiliar with the subject. The second problem is that the knowledge representation structure used may place constraints on the type of knowledge included. The initial problem of actually collecting the knowledge is often overlooked at the start. This function, however, becomes increasingly painful and significant as the project progresses.

A number of different techniques have been developed to minimize these problems. They include simple acquisition systems that 'acquire' their knowledge directly from strictly formatted text files. Micro Expert [Cox82] uses this knowledge acquisition process. More complex systems attempt to guide the knowledge engineer by providing some form of intelligent front-end [Usc84]. The knowledge is then channelled through this acquisition system.

Knowledge acquisition systems vary greatly in sophistication. Simple systems, such as Exsys [Hun85], accept the knowledge but make no attempt to analyze it. Other systems, such as the Teiresias system [Dav82], attempt to understand the knowledge entered and to check

the knowledge base for inconsistencies which may be present.

The Teiresias system is an intelligent front end for EMYCIN, a shell which has developed from the MYCIN project. The EMYCIN system is essentially the MYCIN system without the knowledge base [Buc84]. Teiresias has a number of enhanced features. Not least amongst these is the natural way it attempts to lead the knowledge engineer through the problem of setting out his knowledge base. This system identifies conflicts and similarities in the knowledge base. Teiresias also allows the knowledge base engineer to run parts of his system and to observe the effects. He is able at any time to halt execution and to change the knowledge base. All interactions are in a relatively free natural language format.

Another interesting method of knowledge acquisition is based on concept learning [Sam85]. This method attempts to generalize knowledge as far as possible. For example, assume a system already knows that Mary is a woman and that women are human. If the system is then told that Mary is a bad driver, it may ask if all women are bad drivers. If the answer to this is yes, it could then go on to ask if all humans are bad drivers.

Other systems, such Expert Ease [Cox84], attempt to learn how to solve their problems by induction [Qui79]. These systems, often referred to as learning by example systems, formulate rules by examining a number of different examples and the solutions proposed. The learning pattern used in this process could be compared to that used by law students, who learn the laws by examining the verbatim proceedings of cases which come before the supreme courts. Such law reports contain the evidence led and the judgements handed down and serve as examples for the students.

2.2. The Inference Engine

The inference engine is the heart and mind' of the expert system. It determines the way decisions are made and the general problem solving approach followed. as well as choosing and executing the rules and analyzing the answers. The inference engine also calculates, controls and interprets the certainty factors. which are a function of the certainty values the user associates with his answers.

Broadly speaking. the inference engine follows one of two possible approaches in arriving at solutions; either forward chaining or backward chaining [Wat86]. These concepts become somewhat blurred when dealing with frame based systems. Linked to frames are procedures which are initiated when certain operations are performed on the slots in the frame [Bar81. Wat86].

2.2.1. Forward chaining

Essentially this mechanism attempts to deduce as much as it can with as little knowledge as possible. Forward chaining algorithms can either be very simple or extremely sophisticated. The basic algorithm though remains constant. The system attempts to deduce as much as it can with what it knows until it can deduce nothing further or until it has computed a final solution. If no solution can be found. the system may ask for more information and repeat the process or in the simplest case. state that not enough was known to generate a solution. This mechanism can be thought of as Data Driven'.

The process is complicated by introducing powers of non-monotonic reasoning [McD80. Thi86]. The system would then do more than merely attempt to formulate a solution from what it knew. It would be able to make certain assumptions as well. These assumptions may later be shown to be false. in which case the system would have to undo any decisions which had been made on the basis of the false assumptions.

2.2.2. Backward chaining

The backward chaining approach is 'goal directed' in the sense that the system tries to prove a solution, selected from a list of possible solutions. The mechanism employed first assumes that one of the potential solutions is the solution and then attempts to prove this. If the chosen solution is disproved the system will select one of the remaining possible solutions and attempt to prove it. This process continues until a solution is satisfied or until there are no candidate solutions left. A number of microcomputer systems, including Expert-Ease [Mic84] and Micro Expert [Cox82], use the backward chaining approach.

The advantage of using backward chaining over the forward chaining approach is that the user does not have to specify the facts of the case at the start of the consultation. Requiring this to be done gives rise to two problems. Firstly, the user may not know in what form to present these facts. Of more significance though, he may not know which facts are important and which are not. The backward chaining approach avoids these problems by prompting the user for any information required. The system converts answers to a format which it can use and understand. The backward chaining approach makes no assumptions; the system always asks for information if it does not know the answer.

Considering the limited resources offered in a microcomputer environment, the backward chaining approach is the better of the two as the algorithm is potentially less complex and only the information needed is stored.

2.3. The Interface

The interface section controls the interaction between the expert system and the user. It can be thought of as a post box through which both sides communicate. This section checks the input to ensure that it is in an acceptable form to be used by the inference engine. It also controls the communication with the user by formatting text, acting as a window manager and performing any natural language understanding functions which may be required. Microcomputer based systems. such as Exsys [Hun85] and Micro Expert [Cox82] have relatively unsophisticated interfaces. MYCIN however, has an impressive natural language facility through which users may communicate with the system [Buc84].

2.4. Working Knowledge

The inference engine stores any results which are generated during a session in the working knowledge section. Results of questions asked and rules invoked are remembered for future use during the current session.

The working knowledge section is always empty at the start of a consultation and. as the session progresses. the inference engine will add information to it. This section can be compared to a blackboard in a consultant s room. At the start of an interview the board is erased. As questions are answered and deductions are made, the new information is written on the board. This process continues until a final decision can be made. The consultation is then ended and the board wiped clean. before the start of the next consultation.

2.5. The Problem of Inconsistency in Expert Systems

A problematic issue is the question of inconsistencies. both in the original rulebase and in the answers given in any particular session. Inconsistencies occur in the knowledge base when two rules produce different results using the same conditions [Bra85b]. The problem becomes particularly serious when the knowledge base is too large for a single person to maintain [Dav82]. Rules may be entered at different times and by different people. If these people are not fully aware of the work of the other contributors they could easily enter conflicting rules. An intelligent expert system should therefore be able to analyze its knowledge base and detect inconsistencies of this kind where they occur.

The other side to this issue is that of conflicting user input. In the tax system for example. it is obviously contradictory to state that a person qualifies for the maximum rebate because that person is responsible for the maintenance of a child. and then later. to state that the person has no children. The better expert systems should be able to identify these problems and alert the user to the folly of his ways.

2.6. Conclusion

The system described in the following chapters follows closely the expert system architecture described here. The P.R.O. System is a backward chaining. production system. The structure of the knowledge base is described in chapter 3 and the shell components of the system in chapter 4.

It should always be remembered that the general philosophy behind the design of any expert system shell is that a single mechanism is provided which should be able to handle a diverse set of knowledge bases. In order that an inference engine can operate with a knowledge base, the knowledge base must obviously conform to some predefined structure. At this stage it appears that the technology is not available to create a single inference engine which will drive extremely diverse sets of knowledge bases. The trend has been to design shells which are applicable to particular areas. The P.R.O. System has been designed for use chiefly in the ecological domain.

3. The P.R.O. System Knowledge Base

3.1. Introduction

The previous chapter introduced the important issues in expert system knowledge base design and briefly discussed different knowledge representation techniques. This chapter discusses in detail the structure of the knowledge base used in the P.R.O. Expert System Shell.

It was decided that the knowledge base structure should follow the production system format closely. A number of expert systems have incorporated this format and have shown that it works well. Systems such as MYCIN [Buc84] and Prospector [Dud79] are successful production systems housed on large powerful machines. Expert-Ease [Mic84], Micro Expert [Cox82] and Exsys [Hun85] are good examples of the many production systems available for microcomputers. The proliferation of this style of knowledge representation alone indicates the degree to which researchers feel that it is effective.

A further motivation for adopting the production system format. was that an important aspect of the research would be the development of a simple. effective method of handling uncertainty. In order to make the findings more creditable and more generally applicable. it was decided to follow the familiar production system style.

In practice. it has been found that the knowledge base needs to consist of more than pure rules and questions. The need for general introductory information to the expert system and extra information about rules has also been identified. Additional data on numeric attributes is necessary to check bounds and to assign weightings. The various structures are discussed below and the format and use of each are explained.

In this chapter extensive use will be made of the example tax system,

described in appendix A. for illustrative purposes. This system determines the amount of rebate a tax payer may claim. The amount calculated is based on four basic criteria. namely age. marital status. number of children and number of dependents.

3.2. Introductory Knowledge Structure

```
systemtype(the type of expert system - diagnose or index type
           the name of the expert system,
           introductory remarks to the expert system).
```

Introductory material is provided within the fact 'systemtype', which has three parameters. The first parameter is either the word 'score' or the word 'diagnose', indicating what type of expert system is to be run. The second parameter contains the title of the system in the form of a string. A string in Prolog is enclosed within double quotes (for example, "A tax system to calculate rebates"). The final parameter consists of a list of lists of strings. Lists are delimited by square brackets, such as [["This is a system designed","to ..", ..], ["This system was written by John Bradshaw ..."]]. As strings may not extend from one line to the next, the use of lists enables the knowledge engineer to provide any number of lines of text.

Fig. 3.2.

The systemtype fact should occur only once within a knowledge base. Its main purpose is to tell the inference engine (described in the next chapter) what type of final answer to calculate. Solutions can either be diagnostic or a computed index value calculated from the numeric attributes of the knowledge base. The title is used as a heading for the duration of a consultation.

The facility to include introductory remarks allows the knowledge engineer to provide the user with some background and general information on the system. This introductory information is presented at the start of a consultation. The word processing function within the P.R.O. System. displays each set of strings as a separate

paragraph. This function is described in the discussion of the user interface in the next chapter.

3.3. Rules and Hypotheses

```
hypothesis(name1 and name2,
           the type of answer expected,
           the minimum threshold,
           the maximum threshold,
           the negative threshold,
           the measure of the goodness of the rule,
           an introductory message,
           the answer given on completion of the rule,
           the question text,
           the why, how, explain texts).

rule(name1 and name2
     the answer which the rule will either test or return):-
     the conditions needed to satisfy the rule.
```

name1 and name2 and both single words, for example 'tax' and 'rebate'. The type of answer expected is indicated by either a 'c' or an 'n', representing either a non-numeric or a numeric answer respectively. The three threshold values are integers in the range zero to one hundred. The measure of goodness of the rule is a value between zero and ten. The introductory message is a single string (such as, "Calculating the total tax rebate"). Question, why, how and explain texts are all stored as lists of strings. The 'primary, rebate' hypothesis, in the example tax system, provides a good illustration of the use of this knowledge structure.

In the case of rules, the answer either to be tested or calculated is represented as a more complex parameter. Numeric answers have the form 'n(High value, Low value)' and character answers the form 'c(Single character)'. The structure of the condition part of the rule is very much more flexible than any other portion of the knowledge base. Essentially the conditions consist of references to the other elements of the knowledge base via the use of system defined predicates. The condition 'getvalue(children, rebate, 50, n(ChildHigh, ChildLow))' refers the inference engine to the rule to calculate children rebates. The conditions may also contain any expressions defined by the knowledge engineer, such as those used to calculate the rebate amount.

Fig. 3.3.

The P.R.O. System has adopted the double name as a standard convention for naming all rules, questions, hypotheses, weights and results. By using the name carefully, the knowledge engineer can represent

relationships between facts. For example 'children. number' and dependents. number are unrelated. yet 'person. age' and 'person, married are related as they are both attributes of the person. (This convention was initially implemented while designing an expert system to identify birds. Latin names were used to identify the numerous families and species.)

Each rule consists of two parts. The main part is the 'rule' with its conditions and answer. The answer is returned if all the conditions are satisfied. The other part is a fact, the 'hypothesis', which contains information required by the inference engine when using the rule. Every rule has the facility to hold introductory and answer texts. These strings. which may be empty. are respectively displayed when the rule is entered and if the rule produces a final answer.

A top level rule is a rule which will yield a final answer. In the diagnose' type system. rules of this kind are identified by having top as their first name (name1). In the index type system more than one answer is produced, namely an index for the whole system and the indices for the different components of the system. The rule for the whole system is identified by the names 'top, top'. The rules for the different components have 'top' as the first name and the section name in the second position. Rules which are not top level rules need not have any text in the answer parameter as they will not yield final answers.

The discussion on the inference engine in chapter 4 describes how the answers to the different sections are derived. Chapter 7 differentiates between these two approaches using the RCS system and Aquaculture system for illustration.

The knowledge engineer has the option of attaching a question to a rule. Before the conditions associated with the rule are tested. the system will ask the attached question. This is a short cut method of obtaining the answer which the rule will otherwise compute. The longer method evaluates the conditions in the body of the rule to

compute the answer. This feature has been included because semi-expert users find it frustrating to answer numerous low level, trivial questions to enable the expert system to deduce what they already know and could have told it in the first instance. The problems these users have are often of a higher conceptual nature. They are, therefore, more interested in seeing the problem in a broad perspective, and the global problem solving strategy used by the expert system. Answering simple questions obscures this general methodology.

Associated with questions are three structures which provide the user with explanations. These three parameters operate in the same way as their counterparts in a normal question. These are described in detail in the inference section of chapter 4.

3.3.1. The measure of the goodness of a rule

Within the hypothesis , the knowledge engineer must state how good he thinks the rule is. In every system more is known about some aspects of the problem area than others, and some rules are more thorough than others. This value reflects the confidence which the expert has in the results that will be produced by the rule. If a rule with the highest value (10) succeeds, the system and the user can be sure that the answer is good. Rules with low values indicate that they are likely to produce unsure results.

Consider the example tax system once again. The rule to determine the age rebate is likely to produce the right answer. The only input needed is the person s age and it is highly likely that a person will know his exact age. This rule, therefore, has a rating of 10. On the other hand, the rule to determine the dependent rebates has a rating of 6. The user must understand the wording of the income tax act to determine if his dependents enable him to qualify for a rebate. The confidence which the rule has in its final result is thus lower, as it is not fully confident of providing an accurate answer.

3.3.2. Confidence threshold values

An important aspect of this research has been the handling of uncertainty and the propagation of uncertainty values through a hierarchical set of rules. In the system there are three threshold levels which are used to determine the outcome of this process.

These three threshold levels work in conjunction with the confidence values which the system generates for every rule executed. As the system examines each of the condition parts of the rule, it is not only working towards an answer, but is simultaneously computing a value which reflects the confidence the system has in the final answer. The system maintains two values, representing the confidence in the answer if the rule succeeds and its confidence in the answer if the rule fails. When the rule terminates the system will associate the rule's confidence value with the answer. The precise mechanism used to compute the rule's confidence value is described in detail in chapter 5.

3.3.2.1. Maximum confidence threshold

If the confidence value of a rule at any time exceeds the maximum confidence level, this is an indication that the remaining conditions need not be considered. Enough of the rule has been satisfied, and with a high enough degree of confidence, for the rule to succeed without looking at any of the remaining conditions. If the maximum value is reached, the confidence value which the system associates with the rule's answer is the maximum confidence value, that is 10.

3.3.2.2. Negative threshold

The negative threshold value demarcates the upper limit for the total confidence which the system has in conditions which disprove the rule. For example, the rule to compute the total tax rebate has a negative threshold value of 0. If any conditions in this rule fail with any degree of confidence, the rule will not succeed. By raising this threshold level, the conditions which 'failed' may be prevented from effecting the final outcome of the rule, provided that the negative

threshold is not exceeded. If the rule fails the system attaches the maximum possible confidence value (10) to the result.

3.3.2.3. The minimum threshold

If neither of the two maximum threshold levels are reached before all the conditions have been evaluated. the system will use the minimum threshold value to determine the status of the result. If the confidence value is less than the minimum threshold, the answer will be recorded as unsure. If the minimum threshold value is exceeded however. the system will record that the answer has sufficient confidence.

```
If the negative threshold value is exceeded
     then associate the maximum confidence value with
       the answer and mark the rule as failed.

If the maximum threshold value is exceeded
     then associate the maximum confidence value with
       the answer.

If the negative threshold value is not exceeded and
  if the maximum threshold value is not exceeded
     then if the minimum threshold value is exceeded
               then associate the system confidence value
                    with the answer
          else record the answer as unsure.
```

This pseudo code algorithm summarizes the use and inter-relationships of the three threshold values.

Fig. 3.3.2.

3.3.3. Conditions

The condition part of a rule consists of three main types. 'checkres' is used to check non-numeric results, 'checkvalue' to check numeric results and 'getvalue' to collect numeric values needed in later computations.

```
checkres(name1, name2,
        the relative importance of the condition,
        the non-numeric answer).
```

The non-numeric answer should be supplied if a specific answer is to be tested. for example 'checkres(person, married, 100, c(y))'. This will be true if the person is married, or more correctly, if the answer to the 'person, married' question is yes.

```
checkvalue(name1, name2,
          the relative importance of the condition,
          the bound of a numeric answer,
          the type of numeric comparison to be made,
          the value against which to compare the answer).
```

The bound of the answer is either the high or the low value. Chapter 4 explains that numeric answers are represented as a range and stored as the upper and lower bounds of this range. Depending on the bound to be checked, 'h' or 'l' should be supplied. In some cases the bound to be used is immaterial and the character 'a' (any) should be used. The type of comparison to be made must be taken from the following set - 'eq' (equal), 'gt' (greater than), 'geq' (greater than or equal), 'lt' (less than) or 'leq' (less than or equal). The final parameter is the value against which the bound will be compared. 'checkvalue(person, age, 100, l, gt, 60)' instructs the system to test whether the person's age is above 60.

```
getvalue(name1, name2,
        relative importance of the condition,
        the upper bound of the answer,
        the lower bound of the answer).
```

The last two parameters should be variables, that is, words (or names) with the first letter of each being a capital. If the value is needed later in the rule the variable name can be used. If the value is not to be used again in the rule, but is needed subsequently to enable the system to calculate the score type answer. the variable should be represented by an underscore character ('_').

Again the standard naming convention has been used for these three condition types. The relative importance in all cases is an integer between 0 and 100 (for example, 'checkres(person, married, 100, c(y))') and has the same function as in conditions.

Fig. 3.3.3.

These three conditions reference functions within the inference engine. The remainder of this chapter will be limited to a discussion of the differences between the three condition types. The mechanism used to collect the values is the same for all three condition types. The two names relate the condition either to another rule or to a question which will produce the answers required.

3.3.3.1. Checking non-numeric results

Only 'checkres' may be used to check non-numeric conditions. All parameters must be supplied and the condition will succeed if the question's answer corresponds to that supplied in the condition.

3.3.3.2. Checking numeric values

checkvalue should be used when checking numeric values. All parameters must be supplied. The condition will succeed if the Boolean expression succeeds. Consider as an example the first of the over60. rebate' rules in the example tax system. The condition is 'checkvalue(person. age. 100. 1. gt. 60)' and will succeed if the lower (1) bound of the person's age is greater than (gt) 60.

3.3.3.3. Collecting numeric values

If the values of a numeric attribute are required. 'getvalue' should be used. These values may be required either to calculate the rule answer or to be used later when the system calculates the 'score type index value based on the value of all the numeric attributes. Consider the use of this condition type in the 'child, rebate' rule. The condition reads getvalue(children, number, 100, High, Low)'. High and Low are both used later in the rule to compute the amount of the child rebate allowed.

3.3.4. Relative importance

Every condition has a relative importance or weighting with respect to the other conditions in the rule. The sum of the relative importances in a rule must be one hundred (100). The relative importance of a condition can be conceptualized as the amount of influence the condition has within the rule. The main rule in the tax system - top. rebate - has four conditions each with different relative importances. The primary rebate condition determines the bulk of the final rebate and so has the highest relative importance rating. namely 65.

In different. rules the same condition can have different relative

importance values. These values are determined by the influence the condition exercises in the individual rule. with no regard to its overall importance in the rest of the system.

```
rule(top, rebate, n(H, L)):-
    getvalue(age, rebate, 10, Agehigh, Agelow),
    getvalue(dependent, rebate, 5, Dephigh, Deplow),
    getvalue(primary, rebate, 65, Primhigh, Primlow),
    getvalue(child, rebate, 15, Childhigh, Childlow),
    H = Agehigh + Dephigh + Primhigh + Childhigh,
    L = Agelow + Deplow + Primlow + Childlow.
```

The condition 'primary, rebate' has a much larger value than any of the other conditions as it provides the bulk of the total rebate and thus is the most important attribute. The 'child, rebate' figure is higher than the 'dependent, rebate' relative importance as the rebate per child is greater than that per dependent.

Fig. 3.3.4.

3.3.5. Rules and Turbo Prolog

Besides being syntactically defined by the P.R.O. System interface, the knowledge base also conforms to the specifications of Turbo Prolog.

Rules are legal Prolog clauses. This is advantageous in that it gives the knowledge engineer the freedom and flexibility to recreate exactly the same problem solving methodology as used by the expert. If the expert performs simple calculations. the knowledge engineer can define these directly in a rule. (rules in the example tax system. contain simple calculations). If the calculations are more complex, he can define his own functions to perform them. These functions can then be called from within a rule. Although the knowledge engineer can create his own functions. the three condition types' link the knowledge base and the inference engine. This flexibility was a vital contributing factor to the success of the RCS System. This issue is highlighted again in the discussion in chapter 7.

3.4. Questions

```
question(name1, name2,
        the type of answer expected,
        question text,
        why, how, explain texts).
```

The standard naming convention is once again used. The type of answer expected is indicated by a 'c' or an 'n', corresponding to non-numeric and numeric answers respectively. The four text fields are lists of strings, of the form '["text", "more text"]'.

Fig. 3.4.

The question text contains the question to be put to the user. The why. how and explain texts contain information about the question. Their function is to explain to the user why the question is being asked. how the answer will effect the final result and, in more detail. what the question means. The use of these three fields is discussed in the user interface section of chapter 4.

3.5. Weighting Information for Numeric Attributes

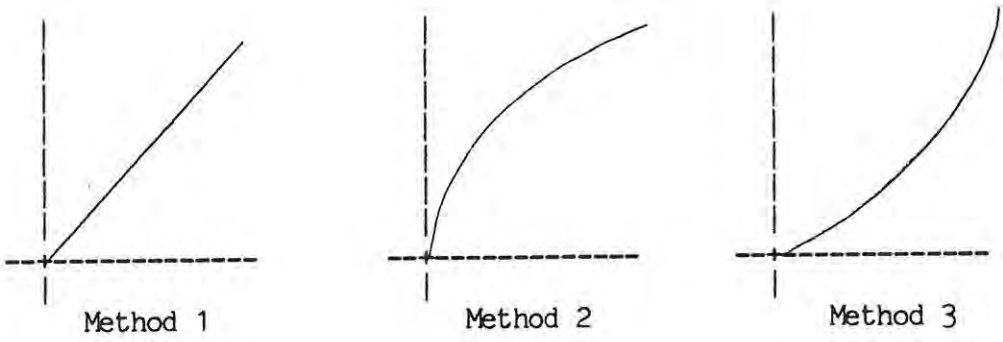
```
weight(name1, name2,
       the range of the answer and the demarcation of
          different categories,
       the weighting factor of the attribute,
       the scaling method to be used).
```

'name1' and 'name2' follow the standard naming convention used. The range is represented by a complex parameter of the form 'class(minimum legal value, 1st bound, 2nd bound, 3rd bound, 4th bound, maximum legal value)'. All bounds are real numbers, while the weighting factor is an integer in the range -30 to 30. Attributes with a deleterious effect are assigned negative weights. The scaling method used is either 1, 2 or 3.

Fig. 3.5.

All questions and rules with questions which yield numeric results require this additional knowledge structure. It contains extra information about the attribute which is used by the inference engine. The values held in the range parameter are used at two stages in the system. At the questioning stage, use of the maximum and minimum values ensures that the answers received are within range. Index type systems use the four intermediate bound values and the weight to determine the contributing factor of the attribute, according to the weighting method specified.

The system uses a percentage of the weight of the attribute in the calculation to determine the final index value. The effects of the three weighting methods is shown in figure 3.5.1.



Different weighting methods for numeric attributes.

Fig. 3.5.1.

Different attributes become significant at different levels. The three weighting methods allow this fact to be represented in the knowledge base. In some cases the effect that an attribute has on the system as a whole is proportional to the amount of attribute present (method 1). In other cases, the difference between the presence of none and a small amount of an attribute is much more significant than the

difference between a lot and a little bit more of the attribute (method 2).

The latter method is used in the case of endemic fish in the RCS System [O'K86a]. An endemic species is found in only one ecosystem in the world. The difference between having none of the species and one present effects the conservation status of the ecosystem greatly. This effect is much greater than that caused by the difference between having five or six of the species.

In other cases the existence of a small quantity of an attribute may be unimportant. The change in the amount of the attribute may, however. become increasingly significant as the amount of the attribute present increases (method 3). If the percentage of water extracted from a river system is low. this attribute cannot be considered a significant feature of the river. As the percentage rises. the significance of this attribute becomes increasingly greater. until such a time as 100% of the water is extracted. This would ensure that the feature becomes the overriding attribute of the whole system. Discussion of a more detailed nature on the attributes mentioned here can be found in the RCS user manual [O'K86b].

3.6. Points to Consider in Designing a Knowledge Base

When designing a knowledge base. a few important points should be carefully checked. All conditions must relate to either a 'rule' or a question . and each rule must be related to a 'hypothesis'. All numeric attributes derived from questions (either 'standard' questions or questions associated with rules) must be assigned an associated weight structure. A single introductory knowledge structure (systemtype) must always be present.

This thesis is not aimed at producing competent knowledge engineers. A few hints are. however. presented. based on the experience gained while developing the two expert systems described in chapter 7. The

RCS (River Conservation Status) and Aquaculture Systems were developed as part of the P.R.O. System project.

To increase the chance of designing a good knowledge base, the knowledge engineer should start with a very simple subset of the whole problem (at most four or five rules). The system should then unfold from here by slowly adding rules and questions to the existing working system. The higher level aspects of the problem should be dealt with first. Lower level issues can be introduced to the system later by converting questions to rules.

The example tax system, which is described in appendix A, illustrates a knowledge base at an early stage of development. Further rules to decide how many dependents entitle the tax payer to an additional rebate could still be added. The pattern illustrated here is to implement a crude outline of the problem first and then to colour in the detail.

3.7. Conclusion

One of the most important aspects to consider when designing an expert system shell is the structure of the knowledge base. If the system imposes too many restraints on the knowledge engineer, he may be unable to represent his problem accurately. The P.R.O. System has been designed to provide the maximum possible flexibility. Some of the variables (such as the threshold levels and relative importances), which are often kept inside the inference engine, have been moved into the knowledge base. The relative importance parameter in the conditions enables the knowledge engineer to specify the most significant conditions in a rule. This feature is not offered in any of the systems reviewed during this research. The free form of the text in the knowledge base has freed the knowledge engineer from the problems of producing a neat end product with full explanations. The incorporation of the Turbo Prolog clause structure has added tremendous flexibility to the whole system.

4. The P.R.O. Expert System Shell

4.1. Introduction

The architecture of the P.R.O. Expert System Shell closely follows the pattern described in chapter 2. The fundamental design approach has been to create the most flexible system possible. without degrading the effectiveness of the system in any way. This has been achieved by including additional information in the knowledge base. which is used by the inference engine to guide it through the knowledge base and to enable it to justify its operations. The knowledge base in this case consists of more than just rules and questions. The additional structures direct the way in which the inference engine handles the more standard components of the knowledge base.

The aim of this chapter is to discuss the workings of the inference engine. the user interface and the working knowledge sections of the P.R.O. Expert System Shell.

4.2. Inference Engine

The order in which the top level' rules are examined is determined by the order in which they appear in the knowledge base. Consequently, the order in which potential solutions are considered is under the direct control of the knowledge engineer. At any point in time, the rule to be chosen is the next one appearing in the knowledge base.

No evaluation is done to find the next best remaining rule. One of the aims of this research is to create a system which will run effectively on a microcomputer. In this environment any expensive evaluation routines become significant with the restricted computational power available. The P.R.O. System (which is implemented in Turbo Prolog) has made use of as many of the routines embodied in the Prolog system as possible. This minimizes the

computational expense of alternative, more sophisticated mechanisms which would be implemented on top of the Prolog system.

The inference engine does, however, follow a high level algorithm. Fundamental to the design of the P.R.O. System is the understanding that answers should not be requested more than once. The system must remember both what it decides itself and what the user tells it. The only proviso is that the certainty value associated with the answer should be above a minimum (3, on a scale of 1 to 10) to justify remembering the answer. Very uncertain answers hold no benefit for anyone. The system, therefore, will ask the same question as many times as are necessary for the answer to be sufficiently certain. Eventually the user will become tired of being asked the same question, and having to admit that he is unsure of the answer. He will, thus hopefully be persuaded to ascertain the correct answer from other sources.

The inference engine first checks if either of the two maximum threshold levels have been exceeded. If this is not the case, it checks if the answer already exists. In the absence of a previous answer, the system will try to ask a question to elicit the answer directly. If all else fails the system invokes a rule to calculate the answer. It is thus essential that every attribute used should be associated with either a question or a rule.

The inference engine is the heart and mind of the expert system. This section manipulates and controls the user interface, the working knowledge and the knowledge base.

The P.R.O. System inference engine can act in two different modes; either as a diagnostic type system or as a system to compute a composite index value of all the contributing numeric data. The first parameter in the systemtype knowledge structure (described in chapter 3) indicates which of the two mechanisms is to be used.

4.2.1. Diagnostic type expert systems

Diagnostic systems are the most common form of expert system. The mechanism involved is easily articulated and is readily transferable to the expert system arena.

The P.R.O. System reports the first solution it finds, although this need not necessarily be the best solution. The user then has the option of instructing the system to look for alternative solutions. If it is important that the best solution be found, the user should force the system to continue until all possible solutions have been evaluated. Once a solution has been found, there is an additional facility which enables the user to change his input values and to observe the effects of the changes on the final results.

The P.R.O. System must obviously distinguish between those rules which produce a final solution and those which do not. All rules whose first name is 'top' are rules which will produce a final solution. The inference engine considers these rules to be the goals required of the expert system and it is the responsibility of the knowledge engineer to 'code' the top rules correctly. The system tries each top rule in the order in which they appear in the knowledge base.

A diagnostic system can also be used to calculate a numeric value to be associated with the top rule's assertion. The example tax system uses the mechanics of the diagnostic system with a single goal. This top rule produces a numeric result enabling the tax system to advise the user whether he may claim a rebate and in positive cases how much he may claim. If there were a number of different ways in which the rebate could be calculated, a number of different 'top rules' could be written to perform these different calculations. The user would then check each goal to find the one which allowed him the largest rebate.

In a diagnostic expert system the results are generated directly by the knowledge base. Systems which compute an index, based on the numeric attributes, operate differently.

4.2.2. Expert systems producing an index figure

The P.R.O. system provides an alternative mechanism to the diagnostic system, namely the index system. This system calculates an index figure based on the combined effect of the numeric attributes in a particular domain. A contributing factor, known as the rating, is calculated for each attribute. These ratings are then used to compute a final score in the range zero to one hundred.

In an index system, the goal is identified by the names 'top, top'. There is only one goal, which should reference subgoals of the form 'top, subgoal name'. In addition to producing an overall index, the system will also compute an index figure for each of the subgoals. The attributes to be included in the subgoal index are recognised if the subgoal name appears as their first name. The exception is the use of the subgoal name rule . The index system will recognize this name and will not attempt to calculate a 'rule' index value.

It was mentioned in chapter 3 that a two name convention could be used to group like attributes. This has proved to be a successful experiment, particularly in the implementation of the index system where more than just the overall index value may be computed.

The rating is calculated according to the mechanism described in chapter 3. Each attribute is assigned a rating according to the class in which the attribute's value falls, the maximum weight of the attribute and the weighting algorithm used (see Fig.3.5.1.). It should be noted that the absence of a deleterious attribute (that is, an attribute with a negative weighting) will produce the greatest possible positive contribution to the final index value.

The final index value is calculated as the sum of the ratings divided by the sum of the maximum possible ratings. For a system to achieve an index value of 100 (the maximum possible), requires the absence of deleterious attributes and the presence of a maximum amount of the

positive attributes in the system.

When using an index system, the knowledge base assumes a rather different function to that when using the diagnostic system. In a diagnostic system, the knowledge base produces the solution directly. The index system uses the knowledge base only to identify those attributes whose values make up part of the final index. Once all the values have been collected, the system calculates the rating using the weight knowledge structure only. At this stage the knowledge base is no longer required and the final computation is performed to produce the index value. In short, the knowledge base in diagnostic systems identifies the answer. In index type systems, it identifies the attributes to be included in the final answer and specifies the relative weight of each of these attributes.

4.3. The User Interface

In chapter 2 it was explained that all expert systems operate by applying a set of predefined rules to a specific set of conditions to arrive at a solution. All expert systems, therefore, require some type of mechanism to elicit the information regarding the specific case from the user. The complexity of interface sections varies widely; from systems, such as MYCIN [Buc84], which support a comprehensive natural language interface, to simple systems, such as Exsys [Hun84], where the scope for user input is very limited.

The approach taken in the user interface for the P.R.O. Expert System Shell is very simple. The user is severely restricted since the system allows only two forms of input, either a single real number or a single character. The main motivation for this very simple and limited system is that the P.R.O. System has been designed to run on microcomputers and is thus subject to the limited resources of these machines. It was felt that the inclusion of even a simple natural language system would not permit the optimum use of the limited computational power available. Furthermore, the number of man hours

required to produce these simple systems (as shown by the length of time. half a man year. required to produce a very simple but comprehensive system to understand simple descriptions of triangles [Pur86]) would have resulted in less effort being devoted to more crucial aspects of the system.

4.3.1. Questions requiring single character answers

Questions requiring single character answers are those questions which can be answered with a yes or no . This type of questioning is necessary to handle those questions which require non-numeric answers. (For example. Are you married in terms of the act?).

The P.R.O. System will ask the user to state the confidence which he has in his answer. This confidence is represented as a value between one (very unsure of the answer) and five (totally confident). This figure is later used by the system to determine the confidence it has in its own results.

If the user does not know the answer to the question. the 'don't know' option applies. A don t know answer has one of two effects. depending on the source of the question. If the question relates to a rule. the don t know answer will force the system to examine the condition parts of the rule and thus compute the answer for itself. If the question is not part of a rule, but a 'normal' question, the don't know' answer will be recorded. The confidence value associated with this answer is zero. that is totally unsure. The system also disregards answers to rule related questions if they are quite uncertain. The justification for this is that if the user is so uncertain. then the contributing factors should be examined and the user s answer discarded. The rule will compute its own answer from the conditions specified in the body of the rule.

The don t know option is potentially very useful. This is particularly true if the system is to be used by people from different backgrounds with a wide discrepancy in expertise. The system tries to

prevent the user from having to supply the lowest level information and allows him to operate at a level where he is most qualified and to which he is suited.

4.3.2. Questions requiring numeric answers

The other type of questions handled by the interface section are those which require numeric answers. These are questions in the 'How much ... and How many ... mold (such as. How many children do you have?).

Numeric questions are simpler to answer than the yes-no variety. All numeric questions require the user to supply a range as the answer (Min and Max). If the two values are the same. the system assumes that the user is totally certain about his answer and will ascribe the maximum confidence to the answer when it calculates its own results. If. however. the answers are different. the system will assign a confidence value which is inversely proportional to the size of the answer range with respect to the maximum possible range. Chapter 5 discusses this algorithm in more detail.

If the user is unsure about either answer. the 'don't know' option may be used. In this case the system takes the corresponding upper or lower bound as the answer. depending on the bound requested. that is either a maximum or minimum value. Once again if the answers to questions linked to rules are considered too uncertain. by virtue of a large divergence in the values supplied. the system will overrule the answers given and attempt to compute a better set of values by examining the condition parts.

4.3.3. Explanations

As described in chapter 3. a set of explanations is associated with all questions. These explanations are provided to keep the user informed about the strategy of the system while it is running. The text supplied should tell the user how his answer will effect the

final result. why the question has been asked and also explain the meaning of the question in more detail. These explanations are aimed at highlighting the decision making methodology used by the expert as portrayed in the expert system.

The explanation routines offered in the P.R.O. Systems are simple to use. yet at the same time they are quite comprehensive. Their effectiveness is a direct function of the thoroughness of the knowledge engineer. The options are invoked by using a letter in the set {w. h. and e}. These are interpreted as requests to the system to explain respectively why the question has been asked. 'how' the answer effects the final result and generally what the question means. The options are available for both questions requiring numeric answers and those requiring single character responses.

4.3.3.1. Why the question has been asked

Users of expert system may become disillusioned with the system if they are unable to understand its problem solving strategy. In the example tax system the user should be told why the system needs to know whether or not he is married.

4.3.3.2. How the answer effects the final result

This option requests an explanation of how different answers to the current question will alter the final result. It is possible that a user might not be too certain about his answer (for example. whether he is considered a married person in terms of the tax act). This explanation should tell him the effect his answer will have on the final result. enabling him to be more careful with his answer if it has a significant effect on the end result.

4.3.3.3. Explain what the question means

If this option is invoked the system should give a detailed explanation of the meaning of the question. For example. a person may not know what kind of marriages are recognised by the tax act for the purposes of calculating rebates. By selecting this option. problems such as this could be resolved. This function can also be used to

explain terms used in the question and refer the user to other sources.

4.3.4. Displaying the knowledge base graphically

The P.R.O. System visually displays all currently active portions of the knowledge base. A set of windows, which contract towards the lower right corner, attempt to give the user a sense of perspective. Each new rule initiates the creation of a new, smaller, 'inner' window. Outer rules thus remain visible and a feeling of purpose is maintained. The user can see how the system moves through the knowledge base and how it links particular pieces of information. This feature is illustrated in figures 4.3.4. and 4.3.6. as well as in appendices B and C.

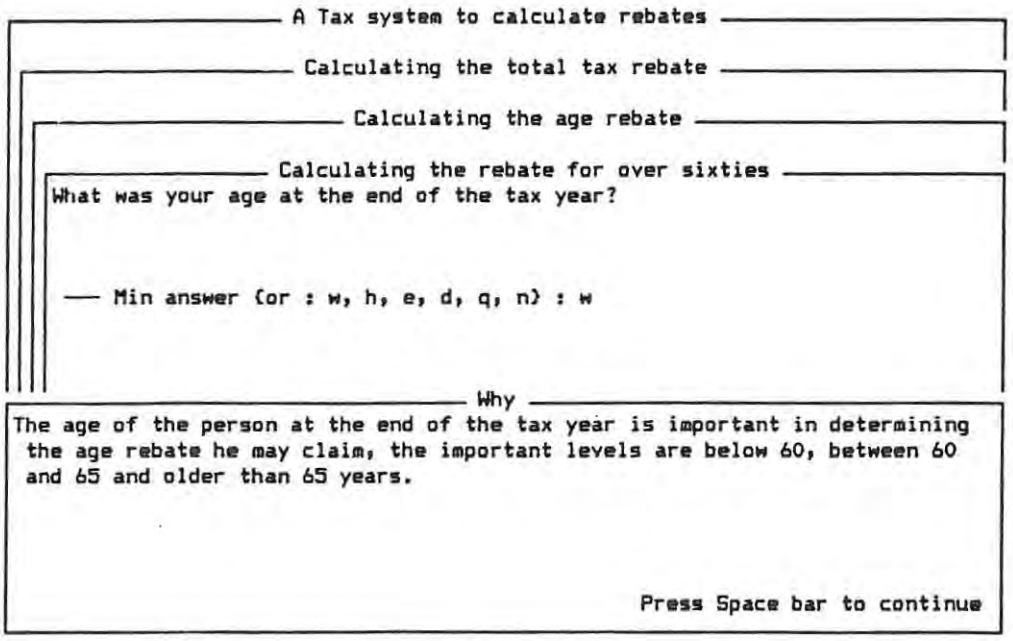


Fig. 4.3.4.

All text displayed on the screen is taken directly from the knowledge

base. It is therefore the knowledge engineer's responsibility to provide the information at a level suitable for the users he anticipates for the system. The format of the text was discussed in chapter 3. A limited word processing facility is available in the P.R.O. System to format the strings neatly on the screen. This feature saves the knowledge engineer from the almost impossible task of trying to produce text which will appear neatly in windows of unknown size. It also ensures that at all times the information is neatly presented. Appendices A and B contain examples of unformatted and formatted text respectively.

4.3.5. Quit and save in the middle of a consultation

An additional option has been provided to leave a consultation without letting it run to completion. The problem with using large systems with many questions is that time may not allow one to complete a whole consultation at a single sitting. This feature allows the user to answer only a portion of the questions and save these results. At the next session the user can load these results and continue answering questions from the point where he left off. Appendix B contains examples of these operations.

4.3.6. Error detection

The pay off for restricting user input is that it becomes relatively easy to provide a comprehensive set of input checks. Numeric answers are checked for range overflows and to ensure that they have been supplied in the correct order. Single character inputs are tested to ensure that they are legal answers. The limited type of answer allowed, makes it possible for the system to give a full diagnosis when errors are detected. Once an error has been detected and reported the expert system will return to the point immediately before the erroneous answer was supplied.

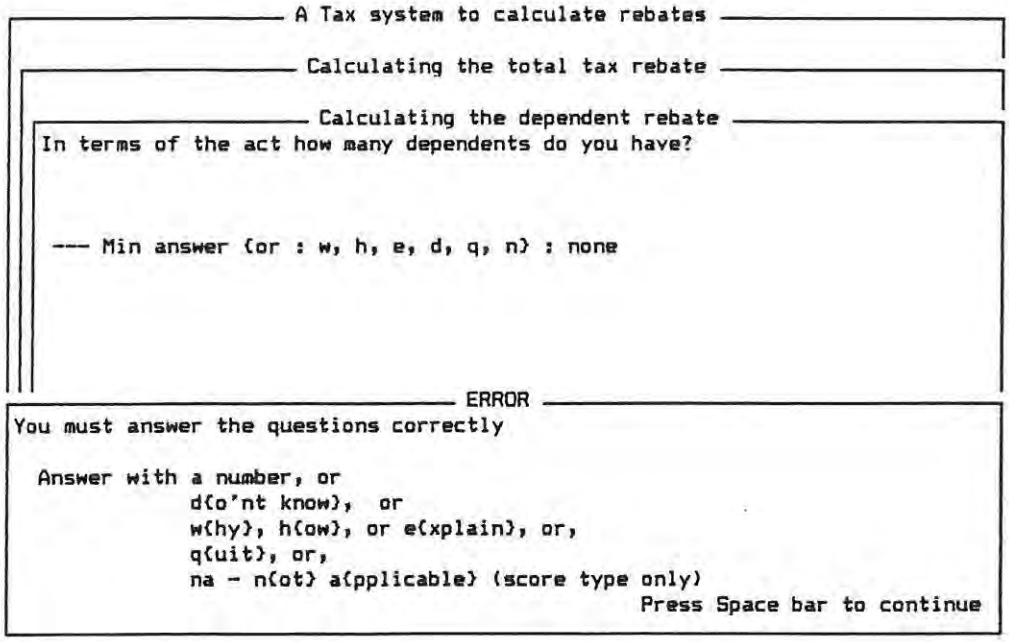


Fig. 4.3.6.

4.3.7. Final answers

Once the system has collected sufficient information to produce a solution it will display the answer text. The option then exists to send the expert system 'back' to look for alternative solutions, if they exist, or to terminate the consultation. At this stage it is again possible to save the given answers for future reference. It is also possible to change certain answers and to observe the effect the changes have on the final solution. It is important to remember that the P.R.O. System will not necessarily produce the 'best' answer first. The system will report the first acceptable answer it finds. The onus is on the user to make the expert system look for other solutions. Only when no other solutions can be found, can one be sure that the best solution has been found. If there are no solutions about which the system is sufficiently confident, it will display

those solutions which were satisfied but which had a low level of confidence.

The results produced by the index system not only give the index values for the whole system and for the subgoals, but also provide full information on every attribute within each subgoal. Figure 4.3.7. below shows the layout of the screen during a session using the RCS system.

The RCS System

Here is a breakdown for the river section

Subsection	Answers Max	Min	Weight	Weighted answers Max	Min	Research Priority	
mainweirs	0	0	-4	0	0	0	10
sewage	2	0	-16	0	-4	0	10
toxic	9	5	-16	-12	-14	0	10
rubbish	1	0	-6	0	-1	12	8
ecosystems	1	0	7	2	0	14	8
siltation	0	0	-5	0	0	0	10
migration	0	0	5	0	0	0	10

The percentage score for the river itself is between:

Min -- 60 and Max -- 69

Press Space bar to continue

This figure shows the format in which subgoal scores are reported. Displayed on the screen is the name of the attribute (that is, the subgoal's second name; the first name being the subsection name), the answers which the user supplied, the weighting of the attribute as defined in the knowledge base, and the relative weighting of each of the answers (in other words the direct contribution of the attribute to the index scores). The last value indicates the need for further investigation of the attribute. This final figure is a function of the importance of the attribute and of the divergence of the answers supplied. Attributes with the highest scores are those which require the most research, and should therefore be considered before those attributes with low values.

Fig. 4.3.7.

4.4. The Working Knowledge Section

In chapter 3 the knowledge base was discussed in some detail. The information in the knowledge base is collected by the knowledge engineer, and is intended to perform a specific task in a specific way as defined by him. The working knowledge section fulfills a different function. This is the area of the system in which temporary values are stored for use during a single consultation. At the end of the consultation the information stored in this section is lost.

The most important information stored in this section consists of results generated by both questions and rules. A set of confidence values of the currently active rules is also maintained for use by the expert system to compute its confidence in the final results.

```
result(name1 and name2,
       the level of confidence in the answer,
       the answer,
       an indication if the condition succeeded or not).
```

The name fields ('name1' and 'name2') follow the same naming convention introduced in chapter 3. The level of confidence is a real number between zero and ten. The answer is a complex term. If the answer is numeric, it has the form 'n(Num, Num)', where Num is any real number. If the answer is non-numeric the term has the form 'c(Char)', where Char is one of 'y', 'n' or 'd' (representing yes, no or don't know respectively). The indication of the success of the condition is a symbol from the set 'true', 'false', 'unsure'.

Fig. 4.4.

The two names identifying the result are taken from the rule or question which generated the result. The level of confidence associated with the answer is either the computed value, where the answer has been computed by the system, or the user's confidence in his answer, if the answer was supplied in response to a question.

The final parameter indicates whether the condition was satisfied or not. All user supplied answers are of course true. As a result this feature is significant only for answers produced by rules. The necessity for this additional parameter is explained by one of the basic philosophies followed in the design of this system. If a condition fails, the system cannot make any decisions on the complement of that condition. The system should therefore, never be in a position to make decisions not based on what the situation is, but based on what it is not. As an example of this philosophy, to say that it is not sunny does not indicate whether it is cloudy or not, since it may be night time!

This approach does result in some inefficiency, but it also does away with several of the problems of conflict which arise when the system assumes certain situations. The problems of non-monotonic reasoning [McD80, Thi86] are thus avoided. This also applies to the problems of

retracting decisions if assumptions are shown to be false.

The inference engine creates, maintains and deletes several other data structures during the normal course of its execution. Although these structures are important to the inner workings of the expert system, they are not fundamental and will not be discussed further in this thesis.

4.5. Expert System Shells, Tools and Languages

The approach which has been taken in the design of the P.R.O. System is one which combines the flexibility of a programming language with the structure of an expert system shell.

Instead of providing the typically strict inflexible environment of an expert system shell, expert system tools attempt to give the knowledge engineer the basic components of an expert system, while at the same time enabling him to 'code' his knowledge relatively freely. Waterman [Wat86] uses the term 'tools for knowledge engineering' and defines these as being "programming systems that simplify expert system development. These include languages, programs and facilities that assist the knowledge engineer" [Wat86]. Jackson suggests that a toolkit should consist of a suite of programs which the knowledge engineer can use as he pleases to construct an expert system [Jac84]. It is envisaged that this suite should include routines to help with all aspects of expert system design and use; from knowledge acquisition routines through to runtime user interfaces and inference systems.

Expert system shells on the other hand show little flexibility outside the areas explicitly catered for by the shell designers. Micro Expert, the first shell to be used in the RCS project, was soon rejected as it did not permit the type of calculations required by the experts [O'K86a].

What has developed as the P.R.O. System could perhaps be termed an expert system tool. The shell title is, however, preferred to prevent the illusion that this system attempts in any way to help the knowledge engineer during the knowledge base design stage. The structure of a shell system has been maintained, while at the same time allowing the flexibility of the use of a full blown programming language in the knowledge base. The P.R.O. System could perhaps be viewed as a shell system which incorporates some of the characteristics of an expert system tool. The flexibility has been the product of the choice of language (Turbo Prolog) and the decision to use the clause structure to represent rules.

Besides the acclaim which Prolog (Programming in Logic) [Clo81, Sam82, Cla84] has received in the Japanese Fifth Generation Project, it has also been extensively used in Britain. The phrase "Logic for expert system" has become common in Britain's IKBS community [Jac84]. Clark and McCabe have been protagonists of these sentiments for a long time [Cla82 and Cla84].

The experience gained while researching this thesis, has certainly shown that the language is admirably suited to the task. The only problems encountered were due to the voraciousness of its appetite for memory, which is a major limitation in the restricted world of microcomputers. The development of microcomputers with the power and size of much larger machines is an on-going exercise. Within a few years the machines of today will most likely look as puny as the machines of the early eighties.

The inference engine and the knowledge base can almost be thought of as two separate systems which share common functions. These functions are supplied by the inference engine to do all the house-keeping and to provide the environment in which the knowledge base can be used.

5. Handling Uncertainties in Expert Systems

5.1. Introduction to the Problems of Uncertainty

Two important aspects of uncertainty arise in the context of expert systems. The first is the type of uncertainty which non-expert users of the system have about a particular situation. The other is the uncertainty which exists as result of the lack of basic knowledge about the mechanics of the particular domain. The effects of both of these lead to uncertainty in any solution; whether this solution has been produced by a computer or by an expert.

Expert systems are required to be able to handle these two types of uncertainty efficiently. The first situation arises when the problem area is clearly understood by the experts, but there is uncertainty on the part of the person providing the system with answers. The latter exists where answers are certain, but where there is a lack of confidence in the effectiveness of the knowledge base. This is a reflection of the expert's uncertainty about aspects of the problem generally. In either of these situations any answer given will contain a degree of uncertainty.

The following example, drawn from the RCS System (described in detail in chapter 7) clearly illustrates these issues. The development of the water resources of the Eastern Transvaal is vital to the future prosperity of the region, indeed of the whole of Southern Africa. The available resources must provide water for the Witwatersrand industrial and urban area, the electricity supply commission's (Escom's) thermal power stations, the rapidly increasing rural population in the region, the Kruger National Park and the people of southern Mozambique. An important survey has been commissioned to investigate the water resources of the region and the likely levels of future usage [NRI86]. The major problems facing this project are exacerbated by the uncertainty surrounding the current relationships which exist between the various influencing factors. There are also

the problems associated with attempts to forecast the future requirements of the region. Questions concerning the population of the area in ten years time cannot be answered with anything more certain than a calculated guess. The existing relationships between the river systems and the Kruger National Park are not fully understood. Besides not understanding the present situation the experts are unable to give confident replies to questions concerning the state of the region in the future. These problems combine both the elements of uncertainty described above.

Expert systems must therefore be usable in situations where there is a lack of certainty on the part of the experts. This has been the case with the RCS system and its designers.

Where such uncertainty exists, expert systems perform a double action. Firstly, they provide a totally unsubjective method of evaluating a set of existing conditions to arrive at an unemotional answer according to a predetermined algorithm. In the second place, they provide a vehicle through which the different experts, each having different priorities, can discuss the problem. The development of an expert system in fact, forces the designers to focus their attention on individual contributing factors, and away from the global problem. This results in the experts being more aware of the holes which exist in their knowledge, which in turn leads to two things. They are able to identify the key areas in which more research is needed and secondly, they are able to discuss the specific issues about which they differ. Initially their differences often appear as major disagreements, however, with further analysis they may reflect a different attitude towards only one or two of the contributing factors.

The other part of the uncertainty problem exists where the user is unsure about some of the factors which are required before any prognosis can be delivered. Expert systems have been used to solve problems of this type. No definitive technique is currently available, which can explain adequately how the uncertain input should

be combined to produce an acceptable, uncertain conclusion.

This raises the following questions. How sure should the user be about his answer before it can be used by an expert system? If there is less than absolute confidence in the input what must be said about the final answer? Need there be a modicum of certainty before any answer can be considered? If so, then at what level should this threshold be placed?

There is a lack of understanding about the way in which experts handle uncertainty and often their uncertain judgements are made on intuitive grounds (or 'gut feelings'). These decisions are particularly hard to substantiate and explain, and the process by which they are derived is difficult to articulate. The need thus arises for a technique which can be implemented effectively in expert systems to provide these systems with a facility for handling uncertainty. This mechanism need not be the same as that utilized by humans, but should nonetheless produce acceptable results.

The emphasis of this thesis has been directed at producing such a system. In the research undertaken, the author has attempted to develop a simple, numeric mechanism [Bun84] which both the domain expert and the knowledge engineer will find easy to understand and acceptable to use.

The method produced will be discussed later in this chapter. In order to place this proposed method in context, some of the more important methods of handling uncertainty which are currently used in some successful expert systems will be considered. The method for handling uncertainty in the P.R.O. System has been designed primarily for use within the restrictions of a microcomputer environment. With this in mind, the methods used in two commonly available, commercial microcomputer expert system shells will be explored.

5.2. Methods of Handling Uncertainty

The two most commonly used methods, namely Bayes' Theorem and the Dempster-Shafer method, have both been used with success in a number of systems. Other systems utilize variations of these methods, in an attempt to make them computationally less expensive. Both methods quantify uncertainty numerically and manipulate these figures to arrive at a final answer. Although there are writers [Mam85] who classify both these methods as statistical models, as opposed to methods which utilize fuzzy logic [Neg85] or non-monotonic logic [McD80, Thi86], they will be treated individually in this discussion.

5.2.1. Bayes' theorem

Perhaps the most well known system which uses the Bayes' theorem, is Prospector [Dud79]. Prospector is an expert system which aids geologists in the analysis of exploration sites for the occurrence of particular ore bodies. The system needs to know what characteristics indicate the existence of particular ore bodies. In order to reflect any uncertainties occurring in the observations, Bayes' theorem is used to calculate the certainty value which the system will associate with its final result.

Bayes' Theorem has the following form:

$$P(C/e) = P(e/C).P(C) \;/\; P(e)$$

where P(C/e) is the probability that 'C' (the conclusion) exists given that 'e' (the evidence) is present.

P(e/C) is the priori probability that the conditions or evidence ('e') will be present given that the conclusion ('C') occurs.

P(C) is the priori probability of ('C') existing.

P(e) is the probability that the evidence ('e') exists.

The same formula is applied whether the evidence consists of a single condition or a number of conditions.

In the event of there being multiple conditions the assumption is made that no inter-relationships exist between the conditions. Bayes' theorem becomes inadequate for situations where relationships are found to exist between contributing conditions [Whi84]. It is unlikely that a very young person will be married or that someone who is married will not have children. The point is, that even in the very simple example tax system (described in Appendix A), relationships already exist between different conditions. White [Whi84] notes that although the problem has been raised before by Adams [Ada76], expert system researchers have largely ignored it.

It appears that in the face of having no better method to use, designers have fallen back on Bayes' Theorem, which is well understood and which has been shown to be correct when the assumptions hold. Indeed, Bramer goes as far as to suggest that workers in artificial intelligence have been content to use methods which they know are not correct, but which to date have produced results which seem to be acceptable and which appear to be correct [Bra85b]. The Prospector system is one of the more successful expert systems written and has shown that answers produced can be good ones. Mamdani et al [Mam85] assert that systems for handling uncertainty that work should all be considered ad hoc "because such techniques have been derived by considerably weakening the theoretical foundations of the original techniques" [Mam85].

Adams [Ada76] explains that methods for handling uncertainty may sometimes appear to present 'correct' solutions when the theoretical implications are that they should not. He suggests that the reason for this is that "it is probable that the model does not founder on [these] difficulties .... because in actual use the chains of reasoning are short and the hypotheses simple". Builders and users of

expert systems are cautioned against using systems which seem to work when the underlying theory suggests that these mechanisms are not applicable to their particular problem. White's rather radical solution to the problem is "to give up all together ... attempting to encode the knowledge of experts directly and instead to derive [the best approximate] model of the domain from a data base of past cases" [Whi85]!

A further problem arises in using the Bayes' theorem. In fields such as medicine where much research has been done, most of the probability figures which the method requires are available. Clinical tests, for example, are conducted whenever new drugs are introduced, and these yield mountains of statistical information. The implementation of the Bayes' theorem in this case, even if non-trivial, is not impossible. In other areas, however, in which the research is limited or not as thorough, it may well be impossible for the expert to produce all the figures which are required for this method to be successful. Ecological applications are examples of such systems. Starfield and Bleloch [Sta83] highlight this when discussing the problems of automating the process of resource management in ecology.

5.2.2. Dempster-Shafer method

An alternative to Bayes' theorem, is the Dempster-Shafer method [Gor85], which decides which is the most likely solution from a set of competing hypotheses. All the evidence need not necessarily be collected at the same time, in which case the system may be able to point to evidence still outstanding, that will have the most discriminatory effect on the set of available hypotheses.

The Dempster-Shafer method assigns an equal value to each of the competing hypotheses at the start of a consultation. Evidence gathered will tend to support particular hypotheses. The complexity of the problem and the computational time needed to compute an answer are therefore, significantly reduced if the size of the hypothesis set is limited to one. The expansion of the hypothesis set leads to an

exponential expansion in the computational time required to resolve the problem [Gor85].

The following example is used to illustrate the workings of this method.

The Aquaculture system [Bru86], which is described in chapter seven, has both Tillapia (T) and Cat Fish (CF) as members of the hypothesis set. If there is evidence (e1) to support T of Ø.5 and further evidence (e2) which supports both T and CF to the degree Ø.75, then the combination of this evidence will combine to support T and CF. The following table shows the resulting situation and the degree to which T, CF and the rest (Ø) of the hypothesis set are supported.

. e1	[T, CF]	Ø
e2 .	Ø.75	Ø.25
[T] Ø.5	[T]	[T]
	Ø.375	Ø.125
Ø Ø.5	[T, CF]	Ø
	Ø.375	Ø.125

The values within the table are calculated by multiplying the associated values outside the table, that is, the supplied values. The sets are decided by taking the union; thus [T] with a confidence factor of Ø.5 and [T, CF] with a confidence factor of Ø.75 produce [T] with a confidence factor of Ø.375 (Ø.5 * Ø.75). The values in the body of the table are then combined (by addition) and used as though they applied to a single piece of evidence, say e3.

e3	[T]	[T, CF]	Ø
	Ø.5	Ø.375	Ø.125

Fig. 5.2.2.

It can be seen that if the set of competing hypotheses is very large and the pieces of evidence numerous, the computational expense will be great. As the number of tables required and the size of the tables increase, so too will the time required to produce a solution

necessarily increase.

A simplified version would be to use the same mechanism, but for systems with only a singleton hypothesis set. Rather than having evidence or rules which produce statements about a number of members in a hypothesis set, such as e1 and e3, there is evidence to support only one hypothesis, e2.

It is important to understand that evidence supporting a particular hypothesis in a set does not give any information about the complement of the set, other than that the remaining evidence resides with the other hypotheses in the set. It cannot be explicitly assumed that if e2 supports [T] by Ø.5, then e2 also states that evidence supporting ~[T] (the complement of [T]) is 1 - Ø.5. The Bayesian systems make this additional assumption [Gor85].

The Dempster-Shafer method is the foundation on which the Confidence Factor method implemented in EMYCIN [vMe84] has been based.

5.2.3. Confidence factors in EMYCIN

The main problem with using the Dempster-Shafer method, as described above, is that, as soon as the system becomes anything larger than a demonstration system, it becomes computationally intractable [Gor85]. Therefore, for the method to be practicable, it needs to be simplified. The designers of EMYCIN's Confidence Factor method have attempted to achieve this [Gor85].

EMYCIN requires that the hypotheses be set out in a strict hierarchy. The system uses only a subset of the hypothesis set at a time. The actual subset used depends on the direction within the knowledge base, in which the investigations are leading the inference system. The designers have also refrained from converting confidence related to the complement of a hypothesis into confidence related to the rest of the set of available hypotheses. Instead, they have recorded this evidence as a general statement about the rest of the hypotheses. The

final step in the method devised, computes the aggregate values of the subsets.

EMYCIN manages to evade the computational expense trap by reducing the scope of the problem being considered at any one time. The designers have emphasized that the success of the system depends very heavily on the way in which the knowledge engineer sets up the knowledge base and the degree to which the full hypothesis set can be split into subsets.

5.3. Microcomputer Based Systems

We now leave the systems which have been implemented on powerful mainframe computers and on Lisp machines, to consider two expert system shells which have been developed explicitly to run on microcomputer systems, namely Micro Expert [Cox82] and Exsys [Hun84].

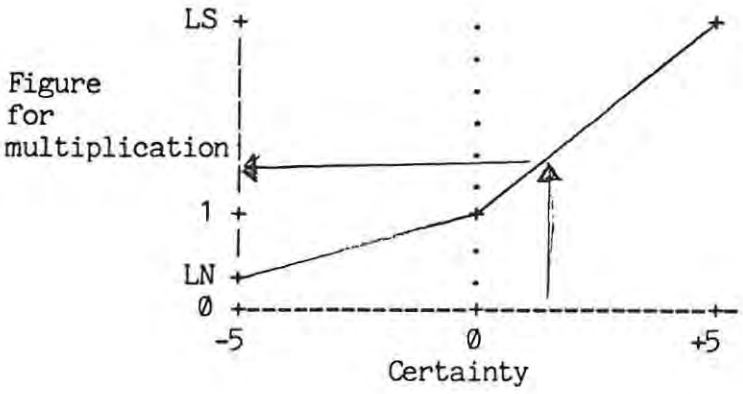
As a result of its limited resources, the microcomputer environment offers a number of additional challenges and problems to designers of expert system shells. Both speed and space are severely limited and it is well known that expert systems are notoriously greedy in respect of both these resources. Any system which includes additional routines for handling uncertainties, over and above the normal mechanisms, should ensure that these routines do not compromise the performance of the system to such an extent that the expert system becomes too painful to use.

5.3.1. Micro Expert

Micro Expert is a simple expert system shell which gives the knowledge engineer a framework and an advice language with which to construct a model of his problem. Each model has a single goal and the system attempts to satisfy this goal and to compute a degree of certainty in its answer. The method which Micro Expert uses to establish this measure of certainty will be discussed here. For a complete description of the system the reader is referred to [Cox82].

The Micro Expert user manual defines certainty as "a measure of truth of the hypothesis" [Cox82]. This can vary from absolutely true to absolutely false, including all the values in between. Although the allusion is given that the method used to determine the confidence in the final goal is Bayesian, it is in fact somewhat different.

Each contributing factor has both a Logical Sufficiency (LS) value (greater than or equal to 1) and a Logical Necessity (LN) value (greater than 0, but less than or equal to 1) and the rule has a priori possibility value. The LS figure is used to calculate the factor used to determine the condition's contribution towards the overall confidence factor associated with a rule, if and only if the condition is satisfied. The LN value is used in a similar process, but in the case where the condition is dissatisfied. The graph (Fig. 5.3.1.) shows how the factor is calculated and how the figure is used. The priori value is converted to an odds value. A priori of 0.1, for example, indicates a 1 in 10 chance of success, or alternatively it represents a ratio of 1 to 9, which when converted to odds yields a figure of 0.1111 (1/9). This figure is multiplied by each condition's contribution factor to produce the certainty figure associated with the result.



Consider the following example:

```
goal Tillapia_ok                        Certainty    Mult. Figure
       fish_ok   LS 50   LN 0.3            4    1+((4/5)*50) = 41
       site_ok   LS 30   LN 0.3           -5              LN = 0.3
       temp_ok   LS 20   LN 0.3            0                 = 1

     priori 0.1 - converted to odds this is 1:9 = 0.1111

actual odds = 0.1111 * 41 * 0.3 * 1 = 1.3665:1

This translates to an actual probability of:

      1.3665/2.3665 = 0.5714
```

The graph is used by first finding the confidence value on the horizontal axis and then moving perpendicularly up until the line is intersected. The value on the vertical axis corresponding to this point of intersection is the figure used to compute the actual odds. The calculation shows how these factors are combined to compute the final odds value.

Fig. 5.3.1.

5.3.2. Exsys

Exsys uses a completely different approach to handling uncertainties. No account is taken of any uncertainty which the user might have in his answers. The only uncertainty which is handled is that which exists in the confidence the rule has in its results.

Exsys uses production rules, which are, however, slightly different to those described earlier in chapter 2. Each rule does not give a single answer, but rather specifies a number of potential answers, each associated with probability values. The system examines all the relevant rules in the knowledge base and generates probabilities associated with each of the different hypotheses. Different probability levels may either exclude or include certain solutions from the potential set of solutions, depending on which of the three available methods is chosen.

The first method is the 0-1 system. This is the simplest system and rules either state that a potential solution is definitely not the

solution (0) or certainly is (1). If a potential solution is assigned the value 0, it is excluded from the final solution set and if it has the value 1, it is included.

The second system is the 0-10 system. This method allows the designer to use the 'grey areas' (1 to 9) which were unavailable to him in the 0-1 system. Once the system has completed its run, it takes the average of all the probabilities collected for each potential solution as the final probability value of the solution. An exception exists if the potential solution at any time received either a 10/10 rating or a 0/10 rating. In either of these cases, this value will 'stick' and the final probability will not be assigned the average, but this value instead.

If three rules give probabilities of 3/10, 5/10 and 7/10 respectively to a particular hypothesis, the final probability of the hypothesis will be 5/10 ((3+5+7)/3). However, if the values were 3/10, 10/10 and 2/10 respectively, then the final probability would be 10/10, signifying that the resulting solution is definitely the solution. If a single hypothesis should be assigned a number of these 'lock out' values, the first value found is the one to be used.

The final method available to Exsys users is the 0-100 system. This works in the same way as the 0-10 system with the exception that there are no 'lock out' values. The values 100/100 and 0/100 are included in the averages, and a solution with contributing values of 30/100, 100/100 and 20/100 would have a final probability of 50/100.

The 0-100 system offers two additional methods of calculation besides the simple averaging procedure. The first regards the values as though they were dependent probabilities. The system takes the individual percentages and multiplies them to calculate the final value. Using the values given above (30/100, 100/100 and 20/100) the final probability would be 6/100. The second alternative handles the probabilities independently and the final solution is then calculated as 1 minus the product of 1 minus the individual probabilities. Using

the same example this would result in a final value of 100/100. As would be expected in a system of this type, if any one rule assigns a value of 100/100, this will be carried through as the final answer by the calculation process.

An advantage of using Exsys is that rules can influence a number of different solutions at the same time in different ways. By stating that the colour of the fruit is green, the system can state (numerically) that the chances that the fruit is actually a white grape or a green apple are good and the fruit could not possibly be a banana. Furthermore, the system is easy to conceptualize. The knowledge engineer of a small knowledge base can easily anticipate what effects his values might have on the final values, except perhaps if he has opted to use either the independent or dependent probability methods. At each stage of the development of the knowledge base, the knowledge engineer therefore knows what effect a new condition will have on the composition of the final solution set.

Disadvantages accrue due to the simplicity of the system. If the knowledge base were to become even moderately large, the knowledge engineer could have problems trying to keep track of all the different rules and their effects on the hypotheses. In such a situation the addition of new rules is likely to produce inconsistencies in the knowledge base. Moreover, the system does not allow the user to state his confidence in his answers, and this greatly reduces the usefulness of any method for handling probabilities. If for example, the user is unsure about all his answers, the system should not return answers with a 10/10 rating, as this does not reflect the true state. It is also unclear from this system what the relative importance of different attributes is. The knowledge engineer is unable to specify the relationship between different conditions.

5.4. The Handling of Uncertainty in the P.R.O. System

A main feature of this research has been the handling of uncertainty. A simple method of propagating confidence factors (Cf) through the rules has been suggested and implemented.

The P.R.O. System tries to move as many of the 'variables' as possible out of the inference engine and into the knowledge base. A combination of the threshold levels described in chapter 3 and relative importance measures have been used. In addition, the knowledge engineer must specify the 'goodness' of each rule. The simple approach for propagating uncertainty through a hierarchical set of rules is discussed in this section.

Although the Logical sufficiency (Ls) and Logical necessity (Ln) values in the P.R.O. System have the same names as values used in Micro Expert, they are in no way related and should not be confused. In fact, it is the author's intention that their meaning should be taken literally. They are used in the same way as the threshold levels described in chapter 3.

Confidence (Cf) values, resulting from questions asked, are obtained using one of two methods. If the answer is either 'yes' or 'no', user is asked for his confidence in his answer. A 'don't know' answer will automatically set the Cf value to zero. Numeric answers are supplied as a range and from this the Cf value is calculated according to the following formula:

Cf = 10 - ((Answer range * 10) / Maximum permissible range)

Once all the conditions have been examined the solution produced by the rule will be assigned a Cf value (called the 'rule Cf' value) according to the following formula:

Rule Cf = ((Ri * Cf's)) / 100 * (Pd/10)

For each condition, a contributing factor is then calculated. This is the condition's relative importance (Ri) multiplied by its Cf value. The summation of these values, divided by one hundred, produces a figure in the correct range. Finally, this value is modified by the expert's confidence in the rule itself (Pd). The resulting figure, in the range zero to ten, is associated with the solution produced by the rule as its Cf value. This 'rule Cf' value can then be used directly in a condition of another rule.

The example illustrated in figure 5.4. is taken from the example tax system and shows how this mechanism works.

```
                          Ri  *  Cf
{primary, rebate}         65  *   8  = 520
{child, rebate}           20  *  10  = 200
{age, rebate}             10  *   2  =  20
{dependent, rebate}        5  *   6  =  30
                                 -----
                                                Pd
Rule Cf = (670/100) * (8/10) = 5,36
```

The above is taken from the rule to determine the total rebate amount. The confidence factor (Cf) of each condition is multiplied by the conditions relative importance (Ri) value in this rule. The resulting figures are summed and the total is divided by 100 to bring it into the right range (0 to 10). The final manipulation is to adjust this value using the 'measure of goodness of the rule' (Pd) value supplied by the expert.

Fig. 5.4.

The resulting value is compared with the threshold values, particularly the Ln value, to decide finally whether the result should be considered successful or not. In this example, the Ln value is zero, the rule Cf value exceeds the Ln value and the result is satisfied.

5.5. Summary of the P.R.O. Mechanism

The P.R.O. System tries to solve some of the problems encountered in handling uncertainty. The model makes no assumptions about the relationships between conditions. For example, if there are three conditions which effect a decision, the P.R.O. System is unconcerned whether they are related or not. What the P.R.O. System does enable the knowledge engineer to represent is the relative importance of each condition in determining the decision. The fact that no assumptions are made about inter related conditions is important in ecological applications, for which this shell is primarily designed. Systems using the Bayesian (or Bayesian based) method [Alv85, Dud79] are hampered in this regard as one of the most fundamental assumptions of Bayes' Theorem is that conditions may not be related.

The P.R.O. System also attempts to provide a mechanism which is easily conceptualized by the knowledge engineer. This is one of the major problems with Micro Expert. In fact, the description of Micro Expert here is not nearly as cryptic as that described in the User Manual [Cox82]. The P.R.O. System tries to avoid too many statistics, as many experts have an innate (albeit sometimes quite irrational) distrust of figures which statistical models produce. This can often be explained by the fact that they find the method too complex and as a result do not understand the full significance of the answers produced or the relationship between the input supplied and the answers returned.

The method used by the P.R.O. shell also attempts to overcome some of the computational problems found in several other methods. The computational expense of the Dempster-Shafer method, for example, expands exponentially as the problem expands [Gor85]. This aspect is especially important in this research where the emphasis has been to design an expert system shell which performs acceptably on a microcomputer. The reason for concentrating on microcomputers, is that the very people who should be using expert systems could find themselves far from large powerful mainframes [O'K86a, Bru86a].

This method also attempts to assign different weightings to different conditions. Other systems use a measure of probability [Buc85, Gor85] and say little about the contribution that individual conditions make towards the final solution. In Micro Expert the knowledge engineer can extend the Logical Sufficiency and Logical necessity values and thereby introduce, in a rather obtuse fashion, some weighting. The P.R.O. System tries to provide a framework in which the knowledge engineer can express the importance of different conditions within a single rule. Taken in conjunction with the careful use of the three different thresholds, relatively insignificant conditions should not on their own, drastically affect the final outcome of the system.

In the ecological environment many of the figures which other statistical models require are simply unavailable [Sta84, Bru86a]. Taking this into account and considering the shortage of explicit, formalized ecological expertise, ecological research is an excellent application area for expert systems. The shortage of accurate statistical information would, however, hinder the effective use of the methods described earlier in the chapter. The P.R.O. System, on the other hand, attempts to capture the more subjective knowledge of the expert for use in the same manner as by the expert himself.

5.6. Dempster-Shafer versus P.R.O.

Three different examples are considered in this section. In each case, the two methods for handling uncertainty will be compared and the different manner in which the results are calculated will be explained in detail. It is assumed that the knowledge engineer has no uncertainty about the rules in each of the examples; in other words, the Pd value in the P.R.O. System is assigned the value 10. This assumption is made since uncertainty of this kind cannot be represented in the Dempster-Shafer model.

5.6.1. Example 1

if A and
B and
C
then R

In the P.R.O. System, the conditions A, B and C have Relative Importances (Ri) of 50, 30 and 20, and are all positively satisfied with confidence factors (Cf) of 8, 6 and 10 respectively. In each case the evidence confirming the hypothesis (R) for each condition (that is, the Ds value), will be calculated as (Ri * Cf)/100. In order to compare the two methods, the same initial evidence has been used in the Dempster-Shafer model. It could be argued that the Dempster-Shafer model would work differently with different input. The author has however, intentionally included the Ri values in the computations of both models so as to exclude any differences this factor might otherwise cause. In the subsequent calculations, the contributing pieces of evidence are numbered e1, e2 and so on.

Condition	Ri	Cf		DS value	
A	50	8	400	0.4	e1
B	30	6	180	0.18	e2
C	20	10	200	0.2	e3
			780		

The P.R.O. System will assign a Cf value of 7.80 to R.

. e1	[R]	Ø
e2 .	0.4	0.6

[R]	0.18	[R] 0.072	[R] 0.108
Ø	0.82	[R] 0.328	Ø 0.492

e4 = R = 0.072 + 0.108 + 0.328 = 0.508

Evidence e1 and e2 are combined to form evidence e4. The values inside the table are obtained from the product of the corresponding values outside the table, for example Ø.Ø72 = Ø.18 * Ø.4. The value of e4 is derived by summing the values supporting R in the body of the table.

. e3	[R]	Ø
e4 .	Ø.2	Ø.8
[R] Ø.5Ø8	[R]	[R]
	Ø.1Ø2	Ø.4Ø6
Ø Ø.492	[R]	Ø
	Ø.Ø98	Ø.394

Belief in R = Ø.1Ø2 + Ø.4Ø6 + Ø.Ø98 = Ø.6Ø6

The belief in R is computed in the same way as the value for e4. At this stage e3 and e4 are the contributing pieces of evidence.

Fig. 5.6.1.

Reference has been made previously to the computational expense of the Dempster-Shafer method. This is already evident in this very simple single hypothesis example.

The P.R.O System requires 2n+2 operations, where n is the number of conditions in the rule. The Dempster-Shafer method on the other hand, requires 8n-6 operations, an increase expense rate of nearly 4ØØ%. This does not include the expense involved in producing the DS value given above.

5.6.2. Example 2

For this example, the rule given in example 1 is used with the same Ri values, but with different Cf ratings. A, B and C are all assigned Cf values of 10, representing the user's total confidence in his answers.

Condition	Ri	Cf		DS value	
A	50	10	500	0.5	e1
B	30	10	300	0.3	e2
C	20	10	200	0.2	e3
			1000		

The P.R.O. System will assign a Cf value of 10.0 to R.

. e1	[R] 0.5	Ø 0.5
e2 .		
[R] 0.3	[R] 0.150	[R] 0.15
Ø 0.7	[R] 0.350	Ø 0.35

. e3	[R] 0.2	Ø 0.8
e4 .		
[R] 0.650	[R] 0.130	[R] 0.5200
Ø 0.350	[R] 0.070	Ø 0.2800

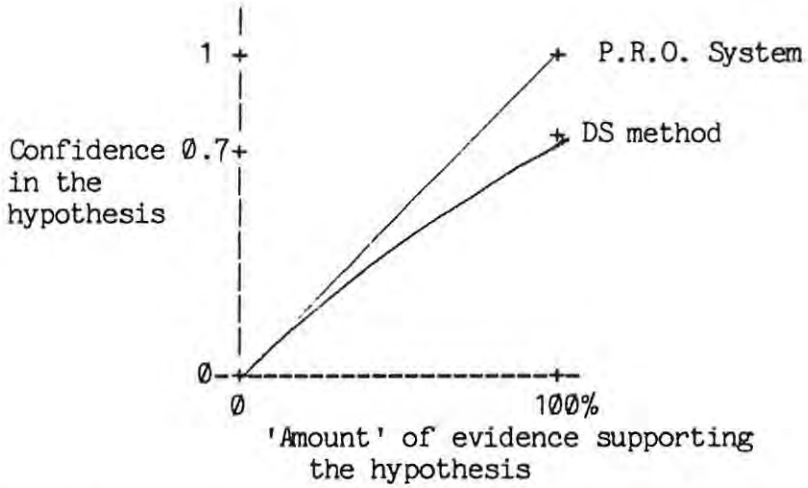
e4 = 0.650

Belief in R = 0.72

Fig. 5.6.2.

It should be expected that if the knowledge engineer is 100% confident about the goodness of his rules and the user is 100% confident about his answers, the system should produce an answer with a 100% confidence rating. The P.R.O. System succeeds in this respect, yet the DS method returns a confidence of only 0.72.

Further research has shown that 0.72 is approximately the upper figure obtainable using the DS model. It varies marginally, in the range 0.70 to 0.75, as the number of conditions varies. The two methods produce the same answer if the user's confidence in the evidence is zero. Figure 5.6. roughly shows the correlation between the two models.



The 'amount' of evidence supporting the hypothesis is a composite figure indicating the combined support for the hypothesis. If all the conditions support the hypothesis 100%, the figure is 100. Conversely, if none of the conditions support the hypothesis, this figure is 0. This 'amount' is very similar to the confidence figure, but its function is to provide a common base for the comparison of the results of the P.R.O. System and those obtained using the DS method.

Fig. 5.6.

5.6.3. Example 3

Condition	Ri	Cf		DS value	
A	50	5	250	0.250	e1
B	15	5	75	0.075	e2
C	10	5	50	0.050	e3
D	10	5	50	0.050	e4
E	10	5	50	0.050	e5
F	5	5	25	0.025	e6
			500		

The P.R.O. System will assign a Cf value of 5.0 to R.

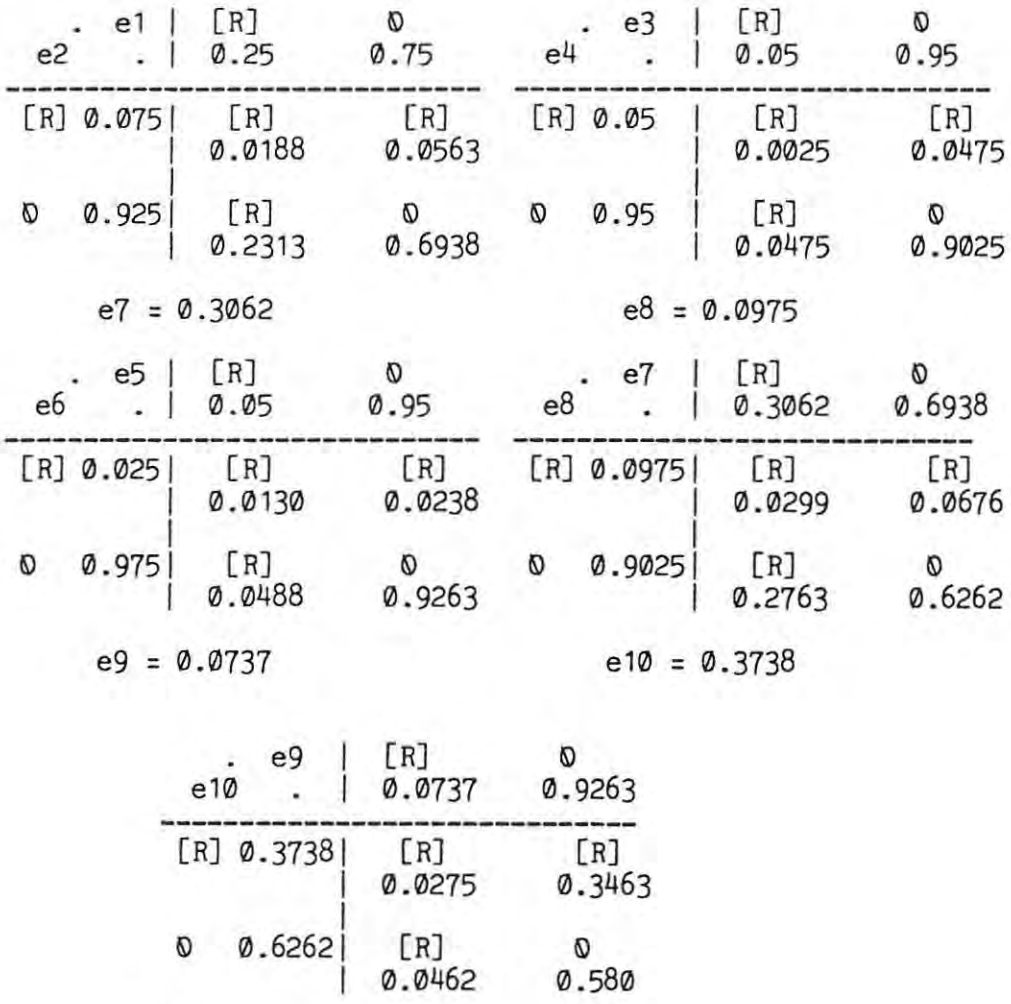


Fig. 5.6.3.

Example 3 further emphasizes the complexity inherent in the DS system compared to the relative simplicity of the P.R.O. System.

5.7. P.R.O. versus other Microcomputer Expert Systems

Both Micro Expert amd Exsys suffer from the same problem. No attempt is made to capture the relative importance of the different conditions directly. Each condition in a rule counts equally towards the final

result. The effect can be achieved partially in Micro Expert by varying the Logical sufficiency and necessity values. The results, however, are not convincing as proven by the following paragraph (5.7.1). (The data used is the same as that given in section 5.6.1.)

5.7.1. P.R.O. versus Micro Expert

For the purpose of the comparison, the P.R.O. relative importance (Ri) figures will be considered equal to Micro Expert's Logical sufficiency (LS) values. The expert's confidence in the rule is the priori value (divided by 10 for Micro Expert). The interpretation of the user's confidence in his answers is the same in both systems, except that the value is divided by 2 in the case of Micro Expert.

Condition	Ri	Cf		LS	Cf	Mult. figure
A	50	8	400	50	4	1+((4/5)*50) = 41
B	30	6	180	30	3	1+((3/5)*30) = 19
C	20	10	200	20	5	1+((5/5)*20) = 21
			780			

The P.R.O. System will assign a Cf value of 7.80 to R.

Micro Expert's figure = 1 * 41 * 19 * 21
= 20559

converted to odds = 20559/20560 = 100%

This figure is obviously not an accurate reflection of the certainty in R. The P.R.O. System is able to reflect the uncertainty which is evident in the conditions.

Fig. 5.7.1.

5.8. Conclusion

One of the aims of this research was to develop a method for handling uncertainty which is both easily understood and explained. It was also intended that the method be efficient in terms of computing resources, while at the same time able to produce acceptable results.

The most commonly used system for handling uncertainties in expert systems is undoubtedly Bayes' theorem or methods based on this theorem. This is particularly the case amongst microcomputer expert systems. Aside from the problems embodied in the system and caused by the initial assumptions, it is not a simple system to explain. It is also a system in which the relationship between the user's input and the system's result are difficult to discern.

The method of handling uncertainty implemented in the P.R.O. System, is acceptable to users at an intuitive level. In an attempt to keep the system's results as transparent as possible to the user, the mechanism employed is very simplistic. This simplicity has also meant that its computational overheads are low. It has been shown that the results produced by the system are related to those derived by the Dempster-Shafer method, which has achieved some measure of success and found support amongst AI researchers concerned with the problems of handling uncertainty. Indeed it appears that the P.R.O. System produces more acceptable results than this method as illustrated in the three examples above. It should be mentioned that the P.R.O. system works well in the context of a simple shell mechanism. If sophisticated techniques (such as non-monotonic reasoning and consistency checking algorithms) are employed, it is uncertain whether this system will perform to the same standards. Development of such a combination is needed to provide experimental information. The author does feel, however, that the P.R.O. mechanism will continue to be useful in these new environments.

In summary, the advantages found in the use and implementation of the mechanism for handling uncertainty as developed for the P.R.O. System,

are simplicity, computational brevity and the ease with which hierarchical combinations of rules and questions can be assembled and altered.

The system is easy to understand and use. This is particularly important when considering users whose expertise may not extend into the fields of statistics and mathematics. When one considers that expert systems are intended to be used by workers 'on the shop floor', it should be anticipated that they may encounter problems with a system even slightly more complicated than the one proposed. It is useless to produce good answers if the user disregards them because of lack of faith in the manner in which they were derived.

The system requires only a few basic computations. The ease with which knowledge engineers can assemble large hierarchies of rules and questions provides one of the plus features of the system. It leaves the knowledge engineer free to consider the problem solving methodology which he is trying to represent in the knowledge base, without having to seriously ponder at every stage the implications of his design on the final confidence factor. In chapter 2 the problems of knowledge acquisition where discussed. It was stated that the process of collecting the knowledge is entirely non-trivial. Any system for handling uncertainty, which alleviates the knowledge engineer's burden, can count this feature as one of its strongest advantages.

6. P.R.O. Knowledge Base Parser

6.1. Introduction

Associated with the P.R.O Expert System Shell is the P.R.O. Knowledge Base Parser. The Parser detects syntactic and (to a limited extent) semantic errors in the knowledge base. No attempt has been made to do any sophisticated checking, such as checking for inconsistencies or for any other type of patterns within the knowledge base.

Experience gained in compiling the knowledge bases for both the RCS (River Conservation Status) and the Aquaculture Systems showed that the experts involved could almost write the knowledge bases themselves. In the process, however, they had problems with the syntax and thus their work had to checked by someone else. The Parser has been developed to allow the experts to check their own knowledge bases. The incorporation of an editor into the system enables the expert to react immediately to any detected errors. The editor has also allowed the system to give a high level of feed back immediately.

The knowledge engineer (the person writing the knowledge base) should initially use any standard text editor to establish his knowledge base. Only then should he put the initial version through the Parser. The Parser detects syntactic errors and will allow him to make the necessary changes. All errors are comprehensively reported in a manner that should leave little doubt as to the exact nature of the problem. Appendix C contains full tracings of the Parser at work, including examples of some of the errors which it detects.

The Parser has two functions. The first of these checks the knowledge base syntax. The second stage rearranges the knowledge base into two files which are then used by the P.R.O. System. This can be thought of as a sort of compilation phase, in which a rearrangement of the text takes place, without any actual compilation as such. The Parser creates two new files. The '.rul' file contains all the rules and any

additional routines which may have been defined. The '.svd' file contains the questions, hypotheses and weight information. The '.rul' file is included in the P.R.O. System at compile-time. The '.svd' file is read into the expert system at runtime.

6.2. The Design Approach

The approach guiding the design of the Parser has been one, that has had to allow for easy and painless changes to be made to the knowledge base when errors are detected. A major design consideration has been the ability to report errors in the most comprehensive and helpful manner possible.

The system has therefore been designed such that the manipulative procedures of the system and those which define the structure of the knowledge base are as independent and distinct as possible. The philosophy followed that of the design of the expert system shell, where the knowledge and the inference processes have been separated.

Apart from checking the syntax of the knowledge base, the Parser also checks the relative importance values in each rule. One of the most basic requirements of the P.R.O. System is that the sum of the Ri values within every rule totals one hundred. This is important if the method for handling uncertainty is to assign the correct confidence value to the rule's result.

6.3. How to use the Parser

Initially the knowledge engineer should use any standard text editor to code his knowledge base. The knowledge base file should be given a 'knb' extension (Eg tax.knb). He may then run the Parser. The Parser will prompt the user for a file name and will create three files. The first is the modified (corrected) version, the old file will be saved with a '.bak' extension. It will also create the two files mentioned

above.

6.4. Conclusion

The Parser system has been found to be extremely helpful, especially for people who have used the system and who were perhaps unfamiliar with the syntax of the P.R.O. Expert System Shell knowledge base. It also ensures that the relative importance values are acceptable.

It is felt that this Parser system could provide the basis for the development of a more sophisticated knowledge acquisition mechanism. The production of a well rounded, user friendly knowledge acquisition system would make the P.R.O. System much easier to use while designing a knowledge base. It is hoped that future research will rise to this challenge.

Examples of the Parser's output are shown in Appendix C.

7. The Development of two Specific Expert Systems

7.1. Introduction

Two expert systems have been constructed using the P.R.O. Expert System Shell. These are the RCS (River Conservation Status) System [O'K86a] and the Aquaculture System [Bru86a]. Discussion in this chapter is based on the papers produced by O'Keffee et al [O'K86a] and those by Bruton et al [Bru86a]. The author was closely involved in both these projects throughout the research reported in this thesis.

The development of the two expert systems was an integral part of the research which culminated in the production of the P.R.O. System. Many of the problems and requirements of these two specific systems prompted the investigation of aspects of the P.R.O. System, such as the need for an efficient method of handling uncertainty.

In the first instance, it was the experts themselves who became convinced that expert system technology provided a means of solving their respective problems. The initiation of the projects by the domain experts was a major factor in the success of the projects. The experts were committed and were prepared to give sufficient time to ensure the success of the tedious process of knowledge acquisition.

The RCS project was started at the same time as research was begun in the developing the P.R.O. System, at the beginning of 1975. An honours student was allocated the project of producing a dedicated expert system [Dan85]. His task was to synthesize the expert's knowledge into rules and questions. The supervision of this project was an aspect of the P.R.O. System research and led to a number of interesting developments, in particular the development of the inference system to produce an index value as its result.

The Aquaculture project was begun a year later. The objective of this project, besides producing an effective aquaculture advisor, was to

test the ideas which had, by then, begun to develop in the P.R.O. System project. This project was chosen as it is substantially different from the RCS System.

It should be emphasized that these projects were both initiated by the domain experts with the sole intention of producing a finished working system. They were made part of the P.R.O. System research effort as they provided the author with a good test bed for his ideas.

7.2. The Design Methodology Used

Many writers [Bar81, Gre83 and Bra85b, Wat86] believe that the most difficult stage in designing an expert system is the knowledge acquisition stage. The approach taken was, therefore, to produce the expert systems by a process of iteration. At the outset the aim was to develop a simple working system as quickly as possible. From there, additions could be made to the knowledge bases, and at each stage the expert was invited to review the results produced thus far and to suggest improvements and extensions.

It has been found that this 'start simple and work up' approach works well. It has the effect of maintaining enthusiasm amongst the experts involved as progress is continually apparent. It also enables the knowledge engineers to grasp the problems and the envisaged final system slowly. An important aspect of both these systems was that the experts were enthusiastic about the projects and thus partook of this time-consuming, iterative process willingly.

7.3. The RCS System

7.3.1. Introduction to the project

The RCS system determines the conservation status of rivers in Southern Africa. For a long time, the conservation of river systems

has received a very low priority [O'K86a]. The flames of industrial development, fanned by the mineral wealth of the region extracts a heavy toll on the natural environment. In addition, the Southern African subcontinent is semi-arid and its rivers contain almost all the exploitable fresh water resources in the region. The river systems have therefore felt the full force of the industrial development, particularly in the Witwatersrand area and its surrounds. The intense pressure for development has often resulted in the destruction of the natural state of the rivers in the region. Consider the Black River in the Cape Peninsular, which today is little more than an open, concrete-lined sewer [O'K86a].

The need for an expert system to analyze the attributes of a river system and to produce an index score of the conservation status of a particular river was clearly apparent. Pristine rivers would receive high scores (maximum 100) and those rivers which had been almost totally destroyed would receive low scores (minimum 0).

The proposed expert system was not intended to halt all further development of the river systems in the region. It was hoped, however, that a system, which embodies the river ecologist's knowledge and decision making process, could appraise a number of different river systems in a completely fair manner. The system could highlight the effects that proposed developments might have on the current conservation status of the river. To show the overall effect of the proposed development, the system could be run with both the current data and with the expected changes in the data. In addition, it was intended that the results produced by the RCS System should be understandable to people other than ecologists.

7.3.2. The system

The project was initiated by the resident expert, who decided that an expert system provided a means to solving the problem. Initial attempts to use Micro Expert [Cox82] proved unsatisfactory. It was found that Micro Expert, a diagnostic system, lacked the flexibility

required to handle the type of system envisaged by the expert.

The P.R.O. System has since been used with success. The RCS System has been approved by all the relevant experts in the country as one which produces good results, which correlate closely with those obtained by experts' research [O'K86a].

The RCS System has exploited the flexibility of the P.R.O. System to the full. An interesting aspect of the system is that large rivers are more likely to have 'more' of some attributes than small rivers. For example, one would expect more different fish species to be found in a large river system than in a small one [O'K86a]. To compute the size of a river, a rule was designed, based on four factors - length, mean annual runoff, catchment size and the stream order of the river. This rule calls several functions which change the weight of particular attributes according to the size of the river.

The project has now reached completion and the river ecologist and his colleagues are satisfied with the results. The system has already been used on two occasions. The first of these was to substantiate a report urging for the conservation of the Okavango river and swamp system [Bru86b], while the second opportunity arose in the evaluation of the water resources of the Eastern Transvaal [NRI86].

7.4. The Aquaculture System

7.4.1. Introduction to the project

Aquaculture in laymen's terms is 'fish farming'. Other, more traditional aspects of the farming industry are well developed in Southern Africa. Aquaculture, however, has long been overlooked. Recently people have become aware of the benefits which can accrue from the development of a successful enterprise. Not only will successful aquaculture ventures improve the menus of the sophisticated restaurants in the urban areas, but the plight of the poorly sustained

people in the rural areas could also be alleviated if aquaculture were to reach its full potential [Bru86a].

With interest rising in aquaculture, experts are becoming increasing hard pressed to attend to the many requests for advice and information [Bru86a]. An expert system was identified as a tool which could have important applications in the area. This project was formulated to give advice to prospective fish farmers on four key factors, namely the species, the culture method, the culture site and marketing information.

7.4.2. The system

Thus far, the Aquaculture system can give advice on the problem of choosing a suitable species according to the existing conditions at the intended culture site. The system to date has knowledge on fourteen potential culture species.

The problem which has hampered development so far is the lack of available knowledge. Much is known about some species (for example Rainbow Trout) and little about others [Bru86a]. This has caused the project to become two-sided. On the one hand, it is a project to translate the aquaculture experts' knowledge into an acceptable form for use in an expert system. In addition though, it has taken on the task of identifying those species which are poorly researched, with a view to stimulating research in these areas.

The system has been designed as a diagnostic type system. To date, there are some fifty rules and as many questions. Although the existing system has not yet been used to advise farmers, it has now reached this stage in its development and will be used in consultations early in 1987. There has already been a positive response from interested people in the aquaculture industry and the experts involved are happy that it is producing good answers [Bru86a].

7.5. Conclusion

The success of both these projects can be attributed to a number of factors. The most important of these is that the experts themselves initiated the projects, played an enthusiastic role, and were keen to see them succeed and to be of practical use and value. Their enthusiasm was an essential ingredient which made the 'start small and work up' approach work well. This in turn led to the effective acquisition of the knowledge required by the systems. A further factor is the ease with which the knowledge in both projects was accommodated by the P.R.O. System. Most of the knowledge acquisition and knowledge engineering work in the Aquaculture System is being done by a ichthyologist who, although computer literate, has had no previous experience in expert system work. The flexibility of the P.R.O. System allowed the RCS System project to be handled neatly.

8. Critical Appraisal

The emphasis of this project has been to design an expert system shell of practical value. The development of the RCS (River Conservation Status) and Aquaculture systems has helped achieve this objective by highlighting the problems involved in producing practical expert systems. Ecological applications were chosen as a target domain of the P.R.O. System because of the availability of interested people and projects in this area. A factor which became obvious at an early stage was the need for such a system to be able to handle uncertainty in an effective way. It also became clear that the intended system should be as flexible as possible. Thus emphasis has been placed on the structure of the knowledge base, one of the most important aspects to consider when designing any expert system. If the structure of the knowledge base is highly restrictive the knowledge engineer will find it very difficult to produce an effective representation of the expert's knowledge.

In order to increase the flexibility of the system, variables which are often kept inside the inference engine have been moved into the knowledge base. The knowledge engineer thus has greater control over the workings of the system. The relative importance parameter and the threshold levels are the most important figures in this regard and their use is one of the novel features built into the P.R.O. System.

The other important design consideration, namely the provision of an effective method of handling uncertainty, has been satisfied by the design of a simple mechanism. This mechanism has been shown to work well and to produce intuitively acceptable answers. It must be emphasized, however, that this success has been achieved within the relatively unsophisticated confines of the P.R.O. System. It is not yet clear whether this method would work as well (though it is believed that it would), in association with more sophisticated inference techniques and mechanisms such as non-monotonic logic and systems to test for inconsistencies.

The P.R.O. System is simple to understand; indeed, so simple that it is possible to explain the workings of the system fully in about ten minutes. Thus the problems which some users have with the answers that expert systems produce are minimized, because the mechanisms which produce the final answer are easily understood. This enables the users to see the relationship between the answers produced and the data which they have supplied.

The mechanism for handling uncertainty is not computationally expensive. Chapter 5 illustrates the few computations required per rule, 400% fewer than is required by a simplified version of the Dempster-Shafer method. This efficiency is once again a function of the system's simplicity. This is especially important to consider in the environment in which this system has been designed to run, that of a microcomputer system. The effective use of microcomputers makes expert system technology accessible to the people who perhaps need it most, that is, the man in the field.

The 'relative importance' associated with each condition enables the knowledge engineer to specify the relative importance of each condition in the rule. Thus the knowledge engineer is able to reflect real life situations in which some conditions are more important than others. This is an important aspect which seems to have been ignored in other systems.

The author feels that the extent to which this system does not rely heavily on accurate statistics has its advantages. In areas of ecology there are problems with providing such values as they are not known by the experts. The system attempts to capture the subjective, or 'rule of thumb', knowledge of the expert as directly as possible. This provides a two fold benefit. In the first place, an expert can supply his knowledge relatively easily.

The second benefit derives from the fact that the user can comprehend the problem solving approach of the expert as represented in the

knowledge base. One of the points highlighted by the two projects implemented using the P.R.O. System is that, once the different experts have isolated the elements which contribute to the total problem, they often realize that their disagreements center on only one or two issues. This reflects a marked change from the initial stance in which they appeared to disagree totally.

This became particularly apparent in the development of the RCS system. At a two day workshop [Mid85] the many experts present were initially extremely sceptical about the validity of the proposed system. After examining the knowledge base, however, they were able to pinpoint their differences. As a result, the system could be strengthened by discussion about specific criteria, without global statements of disagreement and misunderstanding amongst the contributors.

The P.R.O. System has already been put to good use in the development of two practical expert systems. The RCS System determines the conservation status of South African rivers, and has been accepted by all the experts in this country as a system which produces good accurate answers [O'K86a]. Indeed this system is at the moment contributing to a study of the river systems in the Eastern Transvaal and is producing some illuminating and interesting results. The second system is still under development, and recommends a suitable species for aquaculture [Bru86a]. Already the feedback has been most encouraging. A shark identification system (ISIS) has been designed separately, but makes use of the same method for handling uncertainty. The results produced by ISIS correlate with those of the resident expert [For86].

The system still has a few problems. The most serious is the the extent to which the knowledge engineer must become involved in the source of the shell itself. The system would have been much simpler to use if the shell could have been kept entirely separate from the knowledge base right up until runtime. The reason for this inelegance is that Turbo Prolog is not a standard implementation of the language.

Turbo Prolog does not allow clauses and facts to be handled in the same manner. Thus rules, which are also clauses, have to be included in the system at compile-time whereas facts (hypotheses, weights and systemtype) can be consulted at runtime.

Turbo Prolog was used because, apart from this major flaw, it provides a powerful and efficient environment. This is important to ensure that the system be space and speed efficient and therefore be able to operate effectively on a microcomputer. The P.R.O. System was originally implemented in a Prolog system [Spi84] which conforms to the 'Clocksin and Mellish standard' [Clo81]. This version of the P.R.O. System does not suffer from the problems encountered in the Turbo Prolog system. The early version of the expert system shell was, however, abandoned as the particular Prolog implementation proved to be an extremely inefficient user of memory and allowed the development of very simple systems only.

At the outset it was intended that the P.R.O. System should be applied to ecological problems only. The final system has shown that is has sufficient versatility to enable it to be applied to quite disparate topics, such as the example Tax System described in appendix A.

Bibliography

[Ada76] Adams, J., A Probability Model of Medical Reasoning and the MYCIN Model. Mathematical Biosciences, vol 32, pp 177-186, 1976.

[Aik83] Aikins, J., Protypical Knowledge for Expert Systems. Artificial Intelligence, Elsevier, Amsterdam, vol 20, pp 163-210, 1983.

[Alv85] Alvey Directorate, Expert System Starter Pack. NCC, 1985.

[Bar81] Barr, A. and Feigenbaum, E. (eds.), The Artificial Intelligence Handbook. vols 1 and 2, vol 3 (Cohen, P. and Feigenbaum, E. eds.), William Kaufmann, 1981/82.

[Bor86] Turbo Prolog Manual. Borland Inc., California, 1986.

[Buc82] Buchanan, B., New Research in Expert Systems. in Machine Intelligence 10 (Hayes, J. et al eds.), Ellis Horwood, Chichester, pp 269-301, 1985.

[Buc84] Buchanan B. and Shortliffe, E., Rule Based Expert Systems. Addison-Wesley, Reading, Massachusetts, 1984.

[Bun84] Bundy, A., Incidence Calculus: A Mechanism for Probabilistic Reasoning. D.A.I. Research Paper No. 216, Dept. of Artificial Intelligence, University of Edinburgh, 1984.

[Bra85a] Bramer, M. (ed.), Research and Development in Expert Systems. Proc. of the fourth technical conference of the British Computer Society Specialist Group on Expert Systems, Cambridge University Press, Cambridge, 1985.

[Bra85b] Bramer, M., Expert Systems: The Vision and the Reality. Thames Polytechnic, TPL-CIT-85-6, 1986. (also in Research and Development in Expert Systems, Bramer, M. ed.)

[Bru86a] Bruton, M., Dean, N. and Bradshaw, J., An Expert System for Aquaculture. Proceedings Aquaculture '86 conference, CSIR, Pretoria, 1986.

[Bru86b] Bruton, M., The Importance of Conserving the Okavango System. Address given to the Okavango Wildlife Society, 1986.

[Cha85] Charniak, E. and McDermott, D., Introduction to Artificial Intelligence. Addison-Wesley, Reading, Massachusetts, 1985.

[Cla82] Clark, K. and McCabe, F., PROLOG: A Language for Implementing Expert Systems. in Machine Intelligence 10 (Hayes, J. et al eds.), Ellis Horwood, Chichester, pp 455-477, 1985.

[Cla84] Clark, K. and McCabe, F., Micro-PROLOG: Programing in Logic. Prentice Hall International, New Jersey, 1984.

[Clo81] Clocksin, W. and Mellish, C., Programming in PROLOG. Springer-Verlag, Heidelberg, 1981.

[Coh83] Cohen, P. and Grinberg, M., A Theory of Heuristic Reasoning about Uncertainty. The AI Magazine, pp 17-24, Summer 1983.

[Cox82] Cox. P. and Broughton. R.. Micro Expert Users Manual (Version 2.1.1). ISIS Systems Ltd, Redhill, Surry, 1981/1982.

[Dan85] Danilewitz, D., A River Conservation Expert System. Honours project, Dept. of Computer Science, Rhodes University, 1985.

[Dav75] Davis, R. and King, J., An Overview of Production Systems. Stanford University Press, 1975.

[Dav82] Davis, R. and Lenat, D., Knowledge Base Systems in AI. McGraw-Hill, Berkeley, 1982.

[Doy83] Doyal, J., Methodological Simplicity in Expert System Construction. The AI Magazine, pp 39-43, Summer 1983.

[Dud79] Duda, R., Gaschnig, J. and Hart, P., Model Design of The Prospector Consultant System for Mineral Exploration. Expert System in the Micro-electronic Age (Michie, D. ed.), Edinburgh University Press, pp 153-168, 1979.

[Dud81] Duda, R. and Gaschnig, J., Knowledge Based Expert Systems Come of Age. BYTE Magazine, pp 238-278, September 1981.

[ES83] Expert Systems '83. Proc. of the third technical conference of the British Computer Society Specialist Group on Expert Systems, Cambridge, December, 1983.

[For86] Forrester, G., Intelligence in Expert Systems. Honours project, Dept. of Computer Science, Rhodes University, 1986.

[Gor85] Gordon, J. and Shortliffe, E., A Method for Managing Evidential Reasoning in a Hierarchical Hypothesis Space. Artificial Intelligence, Elsevier, Amsterdam, vol 26, pp 323-357, 1985.

[Gre83] Green, R. and Wood, S., Expert Knowledge Elicitation - Or How to Grow Trees. in Expert Systems '83, Cambridge, December 1983.

[Ham84] Hammond, P., Micro PROLOG for Expert Systems. Micro-PROLOG: Programming in Logic (Clark, K. and McCabe, F., Prentice Hall, New Jersey, pp 294-320, 1984.

[Hay83] Hayes-Roth, F., Waterman, D. and Lenat, D. (eds), Building Expert Systems. Addison-Wesley, Reading, Massachusetts, 1983.

[Hay84] Hayes, J., Michie, D. and Pao, Y-H. (eds.), Machine Intelligence 10. Ellis Horwood Ltd., Chichester, 1982.

[Hun84] Huntington, D., The Exsys User's Manual. 1984.

[Jac84] Jackson, P., A Flexible Toolkit for Building Expert Systems. D.A.I. Research Paper No. 245, Department of Artificial Intelligence, University of Edinburgh, 1984.

[Jac85] Jackson, P., Reasoning about Belief in the Context of Advice-giving Systems. Research and Development in Expert Systems (ed. M. Bramer), pp 73-83, Cambridge University Press, 1985.

[Jac86] Jackson, P., Introduction to Expert Systems. Addison-Wesley, Wokingham, England, 1986.

[Jco86] Jacobson, G., The Investigation and Development of a Frame Based Expert System. M.Sc. in progress, Dept. of Computer Science, University of Cape Town, 1986.

[McD80] McDermott, D. and Doyal, J., Non-Monotonic Logic I. Artificial Intelligence, Elsevier, Amsterdam, vol. 13 no. 1-2, pp41-72, 1980.

[Mid85] Two day workshop 'Factors affecting the conservation status of rivers'. Attended by O'Keffee, J., Danilewitz, D. and Bradshaw J, Natal, August 1985.

[Mam85] Mamdani, A., Efstanthiou, J. and Pang, D., Inference under Uncertainty. in Expert Systems '85 (Merry, M. ed.), pp 181-195, Cambridge University Press, Cambridge, 1985.

[Mer85] Merry, M. (ed.), Expert Systems '85. Proc. of the fifth technical conference of the British Computer Society Specialist Group on Expert Systems, Cambridge University Press, Cambridge, 1985.

[Mic79] Michie, D. (ed), Expert Systems in the Micro-Electronic Age. Edinburgh University Press, Edinburgh, 1979.

[Mic84] Michie, D., Expert Ease User Manual. Human Edge Software, Edinburgh, 1984.

[Nil86] Nilsson, N., Probabilistic Logic. Artificial Intelligence, Elsevier, Amsterdam, vol 28, pp71-87, 1986.

[NRI86] Meeting at the NRI. Attended by Dr. J. O'Keffee, Pretoria, October 1986.

[O'K86a] O'Keffee, J., Danilewitz, D. and Bradshaw, J., The River Conservation System - an automated 'knowledge assistant' to help in determining the conservation status of South African rivers. Proceedings of the Conference on Freshwater Wetlands and Wildlife, Charleston, South Carolina, in press, March 24 - 27, 1986.

[O'K86b] O'Keffee, J., Danilewitz, D. and Bradshaw, J., The River Conservation System - A User Manual. CSIR, Pretoria, 1986.

[Pur86] Purchase, H., A Natural Language to a Geometry Tutoring System. technical doc. TD 86/3, Department of Computer Science, Rhodes University, 1986

[Qui79] Quinlan, R., Discovering Rules by Induction from a Large Collection of Examples. in Expert Systems in the Micro-Electronic Age (Michie, D. ed), pp 168-201, Edinburgh University Press, Edinburgh, 1979.

[Rei85] Reichgelt, H. and van Harmelen, F., Relevant Criteria for Choosing an Inference Engine in Expert Systems. D.A.I. Research paper No. 270, Department of Artificial Intelligence, University of Edinburgh, 1985, (also in Expert Systems '85)

[Sam83] Sammut R. and Sammut, C., PROLOG: A Tutorial Introduction. Australian Computer Journal, vol 15, no. 2, pp 42-51, 1983.

[Sam85] Sammut, C., Concept Development for Expert System Knowledge Bases. Australian Computer Journal, vol 17, no. 1, pp 49-55, Feb. 1985.

[Spi84] Spivey, J., University of York Portable Prolog System, release 2. Dept. of Computer Science, York University, 1984.

[Sta83] Starfield, A. and Bleloch, A., Expert Systems: An Approach to Problems in Ecological Management that are Difficult to Quantify. Journal of Environmental Management, vol 16, pp 261-268, 1983.

[Sta84] Starfield, A., Adams, S. and Bleloch, A., A Small Expert System Shell and its Applications. University of the Witwatersrand, 1984.

[Ste82] Stefik, M., Aikins, J., Balzer, R., Benoit, J., Birnbaum, L., Hayes-Roth, F. and Sacredoti, E., The Organisation of Expert Systems: A Tutorial. Artificial Intelligence, Elsevier, Amsterdam, vol 18, pp 135-173, 1982.

[Thi86] ten Thiy, B., A Nonmonotonic Reasoning Expert System Shell with emphasis on an Alternative Algorithm for Reason Maintenance. M.Sc. Thesis, University of the Witwatersrand, Johannesburg, 1986.

[Wat76] Waterman, D., An Introduction to Production Systems. Rand Paper Series, Palo Alto, 1976.

[Wat86] Waterman, D., A Guide to Expert Systems. Addison-Wesley, Reading, Massachusetts, 1986.

[Wei85] Weiss, S. and Kulikowski, C., A Practical Guide to Designing Expert Systems. Chapman and Hall, London, 1984.

[Whi85] White, A., Inference Deficiencies in Rule-based Expert Systems. in Research and Development in Expert Systems (ed. M. Bramer), pp 39-51, Cambridge University Press, 1985.

[Wil86] Williamson, M., Artificial Intelligence for Micro-Computers. Brady Communications, New York, 1986.

[Win77] Winston, P., Artificial Intelligence. Addison Wesley, Reading, Massachusetts, 1977.

[vMe84] van Melle, W., Shortliffe, E. and Buchanan, B., EMYCIN: A Knowledge Engineers Tool for Constructing Rule-Based Expert Systems. in Rule-Based Expert Systems (Buchanan, B. and Shortliffe, E. (eds)), pp 302-313, Addison-Wesley, Reading, Massachusetts, 1984.

Appendix A

The Example Tax System

The tax system described in this appendix is a very simple system designed to illustrate the various features of the P.R.O. Expert System Shell knowledge base.

This system calculates the rebate an individual may claim from the Receiver of Revenue. The calculation of this rebate is based on four criteria - age, marital status, number of children and number of dependents.

A person is entitled to a R120 rebate if he is over sixty, and to an additional rebate of R380 if he is over sixty five. Depending on his marital status, a taxpayer may claim either R620 (single) or R880 (married). A taxpayer may claim R100 for every child and R50 for every dependent.

The system returns two values, the most optimistic and the most pessimistic rebates. The confidence factor returned should give an indication of how good (accurate) the answers are.

The full knowledge base appears on the following few pages. Appendix B contains a few sample screens, taken during a consultation with the expert system.

```
clauses
```

This word must precede all knowledge bases. It is inserted by the P.R.O. Knowledge Base Parser.

```
systemtype(diagnose,
    "A Tax system to calculate rebates",
    [["This is a system designed to calculate tax rebates. Rebates are",
    "calculated according to four categories - Age, Number of dependents,",
    "the marital status of the person and according to the number of",
    "children the person has.",
    "Answers to the questions should",
    "reflect the status at the end of the tax year. ",
    "This is a very simplified version of the system used in South Africa",
    "for the tax year ending 28 February, 1987."],
    ["This is not an accurate system and the result produced should not be",
    "taken seriously. This system was written to demonstrate the features",
    "of the P.R.O. Expert System Shell."],
    ["The system was written by John Bradshaw and is based on the TES",
    "system designed by Martin Steinhobel at Rhodes University."]]).
```

'Systemtype' contains three items of information. 'Diagnose' tells the system that it should invoke the diagnostic section of the inference engine. The title of the system is displayed on all screens during the session. The introductory information is presented at the start of a consultation.

```
rule(top, rebate, n(H, L)):-
    getvalue(age, rebate, 10, Agehigh, Agelow),
    getvalue(dependent, rebate, 5, Dephigh, Deplow),
    getvalue(primary, rebate, 65, Primhigh, Primlow),
    getvalue(child, rebate, 15, Childhigh, Childlow),
    H = Agehigh + Dephigh + Primhigh + Childhigh,
    L = Agelow + Deplow + Primlow + Childlow.
```

Rule 'top, rebate' calculates the total rebate value. It places these results in H and L. This rule calls the age, dependent, primary and child rules to collect their contributing amounts. These rules are called using the 'getvalue' predicate. Note the different relative importances of the conditions. These figures reflect the proportion each attribute contributes to the total amount.

```
rule(age, rebate, n(Agehigh, Agelow)):-
    getvalue(over60, rebate, 50, O60high, O60low),
    getvalue(over65, rebate, 50, O65high,    O65low),
    Agehigh = O60high + O65high,
    Agelow = O60low +O65low.
```

There are two facets to the age rebate calculation. This rule calls the 'over60' and 'over65' rules to determine the total old age rebate allowed. The value is passed back to the top rebate rule as values to the 'Agehigh' and 'Agelow' parameters.

```
rule(over60, rebate, n(120, 120)):-
    checkvalue(person, age, 100, l, gt,    60).
```

This rule states that both 'over60' parameters should get the 'over60, rebate' (R120), since the person's lowest age is above 60.

```
rule(over60, rebate, n(120, 0)):-
    resetrule,
    checkvalue(person, age, 100, h, gt, 60).
```

This alternative version of the 'over60' rule is invoked if the condition of the previous rule did not hold; in other words, if the lower bound of the person's age is less than 60. This rule checks to see whether the upper bound is above 60. If this is the case, only one parameter (the high value) receives the rebate. 'Resetrule' instructs the system that in the event of this rule being satisfied, only the confidence factor associated with this rule should be associated with the answer. If the system was not told this, the previous rule's confidence factor would impact on the result.

```
rule(over60, rebate, n(0, 0)):-
    succeedrule.
```

If none of the previous 'over60' rules apply, no over 60 rebate may be claimed. 'Succeedrule' tells the system that this rule succeeds with absolute certainty.

```
rule(over65, rebate, n(380, 380)):-
    checkvalue(person, age, 100, 1, gt, 65).

rule(over65, rebate, n(380, 0)):-
    resetrule,
    checkvalue(person, age, 100, h, gt, 65).

rule(over65, rebate, n(0, 0)):-
    succeedrule.
```

The 'over65' rules work in exactly the same manner as the 'over60' rules.

```
rule(dependent, rebate, n(Dephigh, Deplow)):-
    getvalue(dependents, number, 100, High, Low),
    Dephigh = High * 50,
    Deplow = Low * 50 .
```

The dependent rebate is calculated by allowing R50 for each dependent. The system ascertains the number of dependents by asking a question.

```
rule(primary, rebate, n(880, 880)):-
    checkres(person, married, 100, c(y)).

rule(primary, rebate, n(620, 620)):-
    succeedrule.
```

If the result of the married condition is true, a rebate of R880 is allowed; otherwise a rebate of R620 may be claimed. The hypothesis associated with this rule (see further down) has a question. The primary rebate may thus be determined by answering this question, or if the user is unsure of the answer, by answering the married question.

```
rule(child, rebate, n(Childhigh, Childlow)):-
    getvalue(children, number, 100, High, Low),
    Childhigh = High * 100,
    Childlow = Low * 100.
```

This rule works in exactly the same way as the dependents rule.

```
hypothesis(top, rebate, n, 0, 100, 0, 10,
    "Calculating the total tax rebate",
    ["The total rebate which you may claim is:"],
    [],
    [],
    [],
    []).
```

This hypothesis relates to a rule which produces a final solution and thus has an answer ('["The total ...]'). No question is associated with this rule and so the last four parameters are empty lists. All rules must have introductions ("Calculating..."). In the top line the 'n' tells the system to expect a numeric answer. The first 0 is the logical necessity figure and the confidence in the rule's answer must exceed this for the rule to succeed. The 100 is the logical sufficiency value. If the rule's confidence in its answer at any time exceeds this value the rule will halt and succeed immediately, disregarding any remaining conditions. A value of 100 can never be exceeded, and therefore this figure ensures that all conditions are evaluated. The second 0 is the negative threshold. If at any time the rule's confidence that it will fail exceeds this figure, the rule will fail immediately. The value of 0 ensures that as soon as any condition fails the entire rule will fail. In the case of a rule, this figure is unimportant as all conditions are of the 'getvalue' type which never fails. The final figure is the 'measure of goodness' of the rule. A value of 10 indicates that the knowledge engineer has full confidence in the result produced by this rule.

```
hypothesis(age, rebate,  n, 0, 100, 0, 10,
    "Calculating the age rebate",
    [],
    [],
    [],
    [],
    []).
```

```
hypothesis(over60, rebate, n, 0, 100, 0, 10,
    "Calculating the rebate for over sixties",
    [],
    [],
    [],
    [],
    []).

hypothesis(over65, rebate, n, 0, 100, 0, 10,
    "Calculating the rebate for over sixty fives",
    [],
    [],
    [],
    [],
    []).

hypothesis(dependent, rebate, n, 0, 100, 0, 6,
    "Calculating the dependent rebate",
    [],
    [],
    [],
    [],
    []).
```

The discussion of the 'top, rebate' hypothesis is applicable to the above four hypotheses, with the exception that these rules do not yield final solutions and so the hypotheses contain no answer texts.

```
hypothesis(primary, rebate, n, 0, 100, 0, 8,
    "Calculating the primary rebate",
    [],
    ["How much primary rebate can you claim?"],
    ["In order to calculate the total rebate the system must calculate",
    "the primary rebate which the person may claim."],
    ["Depending on whether or not the person is a married person in terms of",
    "the act they may claim either R620 or R880 primary rebate."],
    ["A person may either claim R620 if he or she is unmarried, or may claim",
    "R880 if he or she is married in terms of the act.  The act states that a",
    "person is married if he or she is married in terms of any law of custom",
    "and is living permanently with his spouse, or if the person is a",
    "widow or widower, or if the person qualifies for a child rebate and",
    "is responsible for the maintenance of the child."]).
```

This hypothesis illustrates the type of information held if a question is associated with the rule. Four lists of text are provided. The first of these is the question, the other three are why, how and explain texts respectively. The 'measure of goodness' of the rule is only 8 as the knowledge engineer was unsure of the answers which it might give. This was mainly due to the problems which the user might

have in interpreting the act correctly. The figure is less than the corresponding figure in the child rebate rule as the amount involved per dependent is smaller.

```
hypothesis(child, rebate, n, 0, 100, 0, 6,
    "Calculating the child rebate",
    [],
    [],
    [],
    [],
    []).
```

Nothing new can be said about this hypothesis.

```
question(person, age, n,
    ["What was your age at the end of the tax year?"],
    ["The age of the person at the end of the tax year is important in",
    "determining the age rebate he may claim, the important levels are",
    "below 60, between 60 and 65 and older than 65 years."],
    ["If the person was older than 60 at the end of the tax year he may claim",
    "R120. If the person was older than 65 then the rebate would include an",
    "additional R380."],
    []).
```

The order of the question, why, how and explain texts are the same in questions as they are in hypotheses. The 'n' indicates that the answer to the question is numeric. No explanatory text is supplied.

```
question(dependents, number, n,
    ["In terms of the act how many dependents do you have?"],
    ["The number of dependents are an important variable in determining the",
    "amount of rebate a person is entitled to."],
    ["A taxpayer may claim R50 for every dependent recognised by the act."],
    ["In terms of the act a person may be classed as a dependent if the",
    "person was less than 18 years old at the end of the tax year, if",
    "the taxpayer spent more than R200 on the persons upkeep and if the",
    "person is not a child or stepchild of the taxpayer. A person may also",
    "be classed as a dependent if the person is incapable of maintaining",
    "himself and the taxpayer spent more than R200 in maintenance."]).
```

```
question(person, married, c,
    ["Are you married in terms of the act?"],
    ["The amount of primary rebate and thus the total rebate will differ",
    "if the person is a married person in terms of the act or not."],
    ["If the person is a married person in terms of the act they will be",
    "allowed a rebate of R880 otherwise the receiver will allow them a",
    "rebate of R620."],
    ["The act states that a person is married if he is married in terms",
    "of any law of custom and is living permanently with his spouse, or if",
    "the person is a widow or widower, or if the person qualifies for a",
    "child rebate and is responsible for the maintenance of the child."]).
```

The 'c' indicates that a 'yes-no' type answer is required for this question.

```
question(children, number, n,
    ["How many children do you have?"],
    ["The number of children which the person has effects the amount which",
    "the receiver will grant him as a tax rebate and thus an important",
    "factor in determining the final amount."],
    ["The rebate for children recognised by the act is R100 for each child."],
    ["The act states that a person can be regarded as the taxpayer's child",
    "for the purposes of calculating rebates if that person is either a",
    "natural child, a stepchild or an adopted child and if the person was",
    "under the age of 18 at the end of the tax year and was alive during",
    "the year or if the person was under the age of 26 at the end of the",
    "year and was a full time student during the year and was dependent",
    "upon the taxpayer for financial support."]).

weight(primary, rebate, class(620, 620,620,620,620, 880), 0, 0).

weight(person, age, class(0, 0,0,0,0, 130), 0, 0).

weight(dependents, number, class(0, 0,0,0,0, 10), 0, 0).

weight(children, number, class(0, 0,0,0,0, 10), 0, 0).
```

As this is a diagnostic system the only figures which are important are the first and last in the class parameter. These indicate the maximum legal bound for these numeric answers. If this were an index type system, values in between would break the range into five classes. The last two figures would indicate the weight of the attribute and the weighting mechanism to be used in determining what quantity of the weight should effect the final index value.

Appendix B

Appendix B contains tracings of the P.R.O. Expert System Shell.

These tracings have been produced using the example tax system which was described in appendix A. The tracings are screen dumps. They are intended to give the reader an indication of the workings of the P.R.O. System.

Two consultations are shown here. The first consultation consisted of answering all the questions.

```
Age                          : 26 (Both answers)
Amount of primary rebate     : d  (Both answers)
Number of dependents         : 0  (Both answers)
married?                     : n  (Confidence 5)
number of children           : 0  (Both answers)
```

At the end of the consultation the results were saved in "myself".

The next consultation started by loading this file of results in and changing the married answer to yes. The new results are given.

The workings of the system to produce an index figure are the same as those illustrated for the tax example. The results, however, are reported differently. At the end of this appendix, a few screens are given, showing the type of results produced by the RCS System.

Introduction to the P.R.O. Expert System Shell

Welcome to the P.R.O. Expert System Shell

The P.R.O. system has been developed by John Bradshaw in the Department of Computer Science, Rhodes University, Grahamstown, South Africa

(c) The P.R.O Expert System Shell - John Bradshaw

Press Space bar to continue

A Tax system to calculate rebates

Welcome to A Tax system to calculate rebates

This is a system designed to calculate tax rebates. Rebates are calculated according to four categories - Age, Number of dependents, the marital status of the person and according to the number of children the person has. Answers to the questions should reflect the status at the end of the tax year. This is a very simplified version of the system used in South Africa for the tax year ending 28 February, 1987.

This is not an accurate system and the result produced should not be taken seriously. This system was written to demonstrate the features of the P.R.O. Expert System Shell.

The system was written by John Bradshaw and is based on the TES system designed by Martin Steinhobel at Rhodes University.

Press Space bar to continue

```
──────────── A Tax system to calculate rebates ────────────
 ──────────── Calculating the total tax rebate ────────────
  ──────────── Calculating the age rebate ────────────
   ──────────── Calculating the rebate for over sixties ────────────
   What was your age at the end of the tax year?


    --- Min answer (or : w, h, e, d, q, n) : w


 ──────────── Why ────────────
The age of the person at the end of the tax year is important in determining
 the age rebate he may claim, the important levels are below 60, between 60
 and 65 and older than 65 years.


                                                Press Space bar to continue
```

```
──────────── A Tax system to calculate rebates ────────────
 ──────────── Calculating the total tax rebate ────────────
  ──────────── Calculating the dependent rebate ────────────
  In terms of the act how many dependents do you have?


   --- Min answer (or : w, h, e, d, q, n) : q


  Are you sure you want to quit (and save)  this consultation (y/n) :
```

A Tax system to calculate rebates

Calculating the total tax rebate

Calculating the dependent rebate

In terms of the act how many dependents do you have?

-- Min answer (or : w, h, e, d, q, n) : e

Explain

In terms of the act a person may be classed as a dependent if the person was less than 18 years old at the end of the tax year, if the taxpayer spent more than R200 on the persons upkeep and if the person is not a child or stepchild of the taxpayer. A person may also be classed as a dependent if the person is incapable of maintaining himself and the taxpayer spent more

Press Space bar to continue

A Tax system to calculate rebates

Calculating the total tax rebate

Calculating the dependent rebate

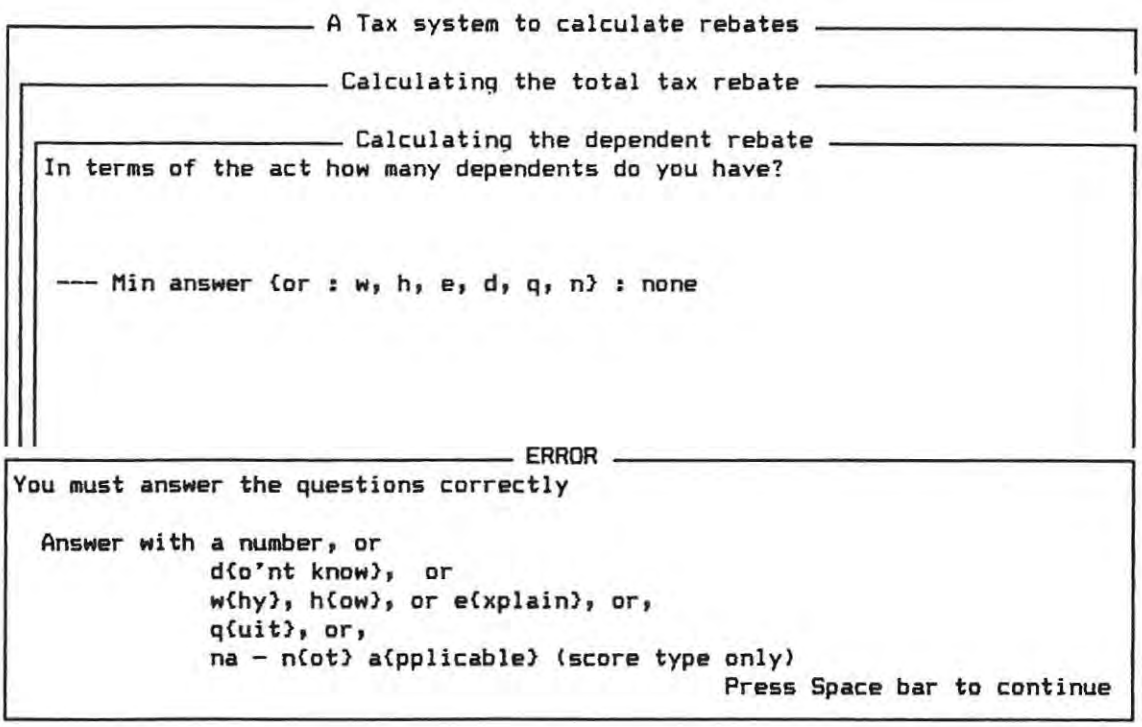
In terms of the act how many dependents do you have?

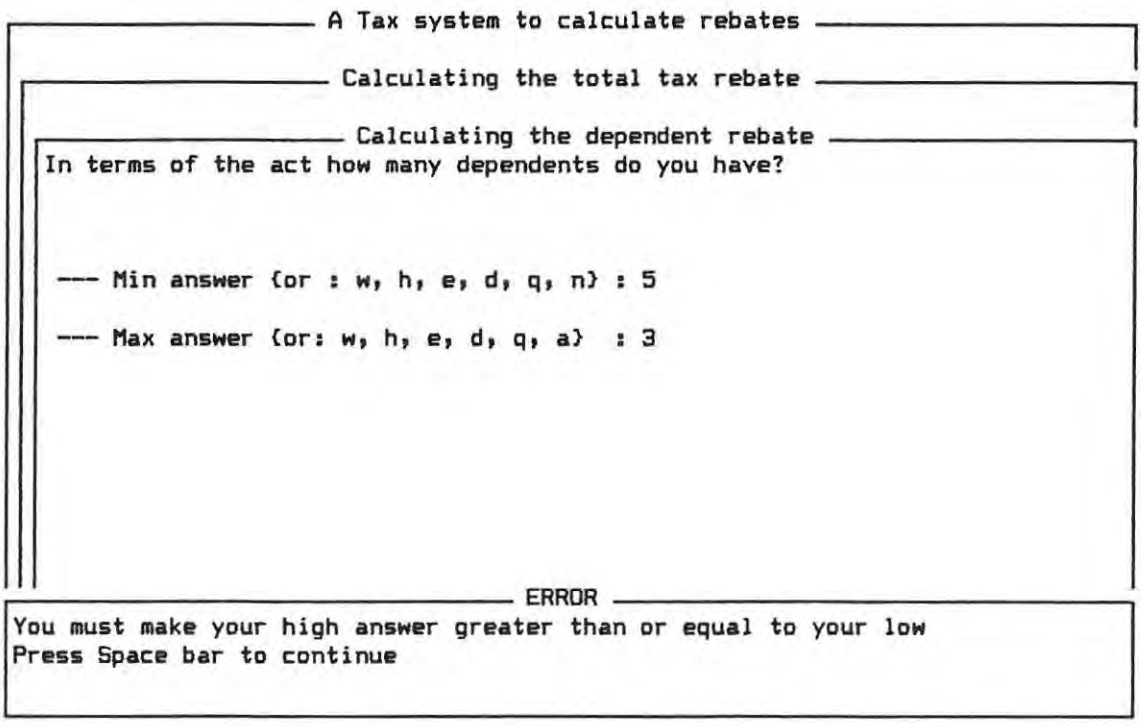
--- Min answer (or : w, h, e, d, q, n) : e

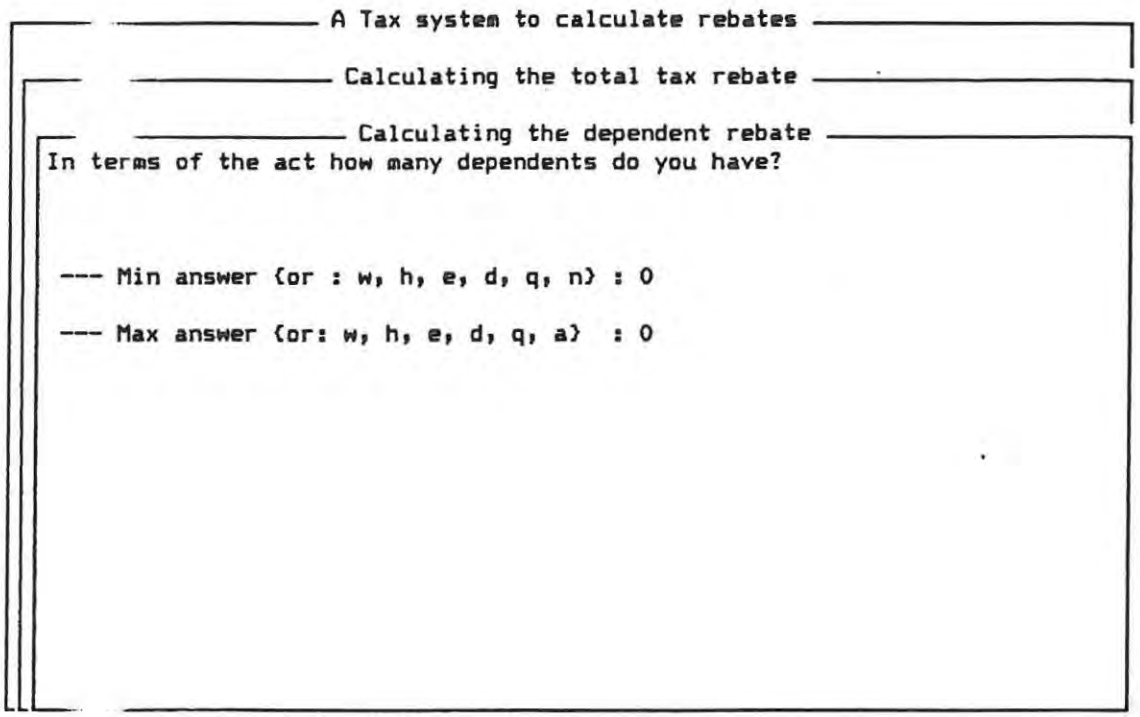
Explain

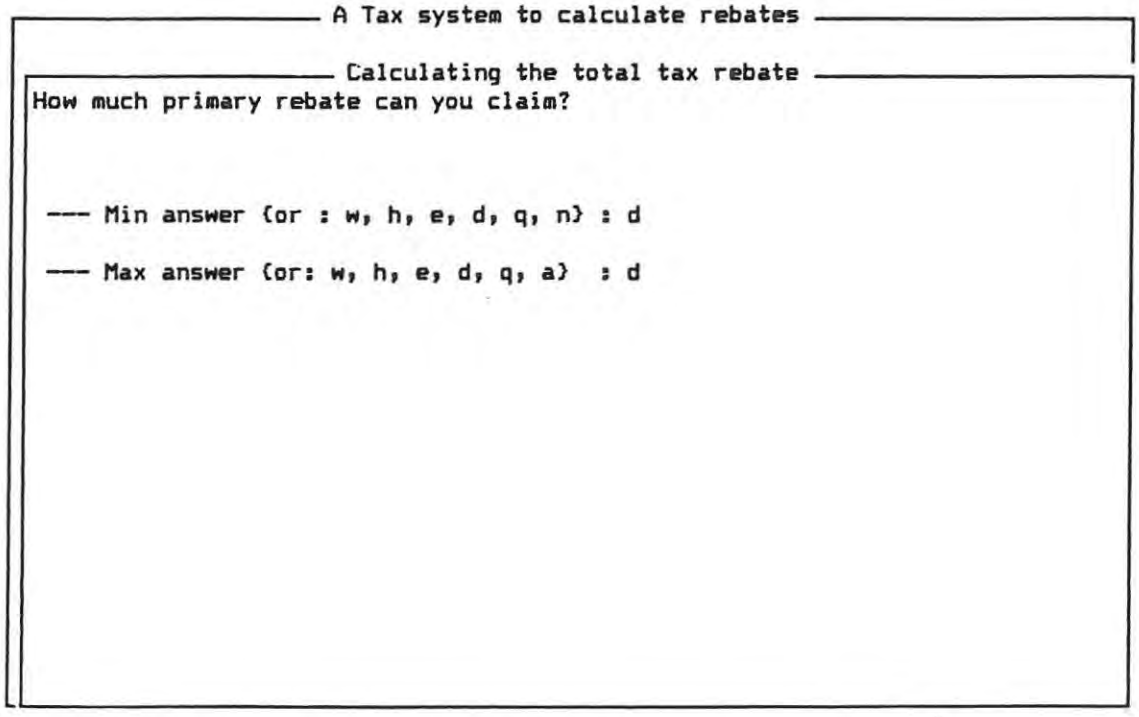
than R200 in maintenance.

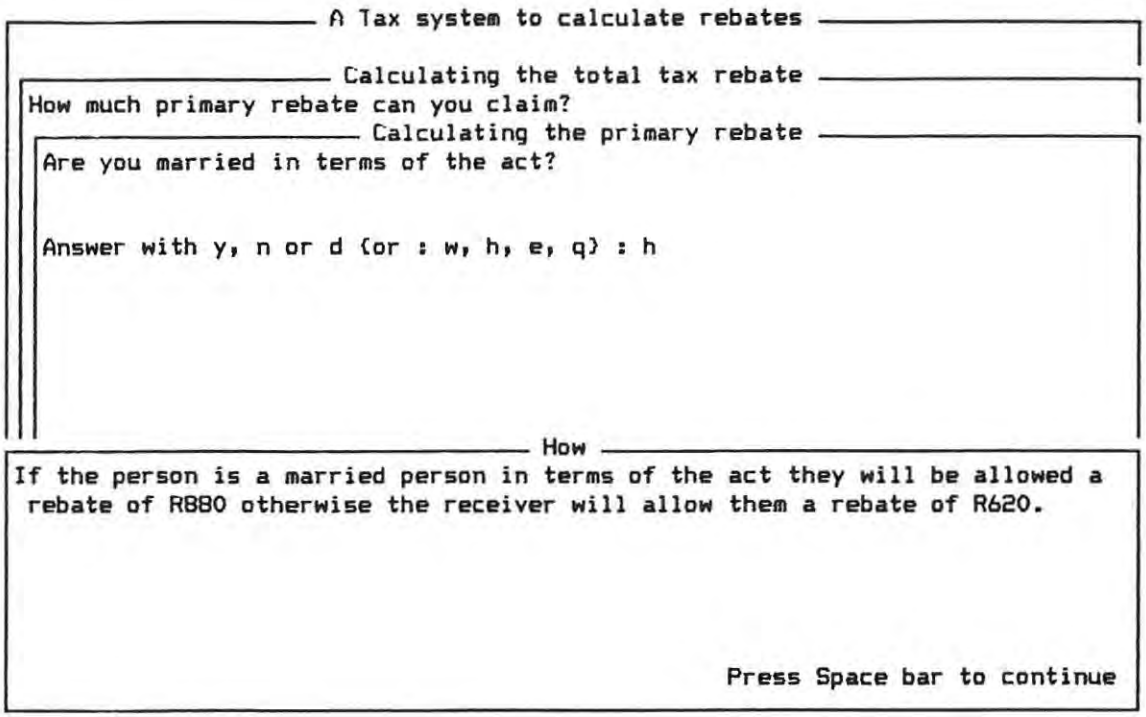
Press Space bar to continue

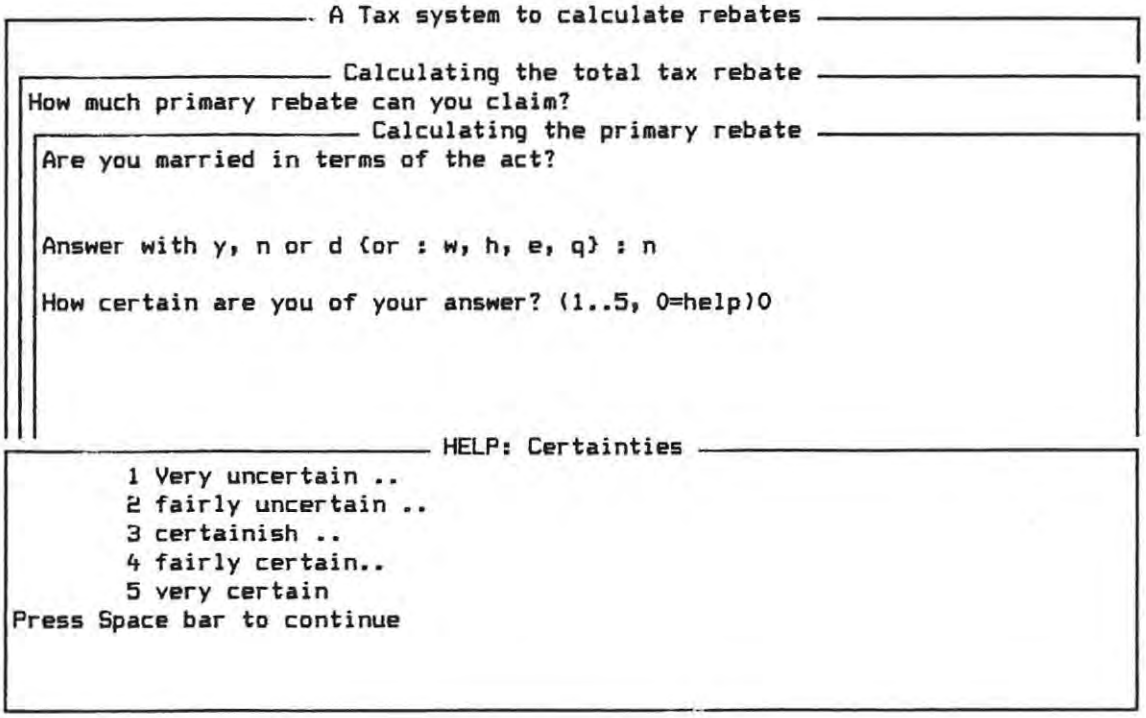

A Tax system to calculate rebates
Calculating the total tax rebate
Calculating the dependent rebate
In terms of the act how many dependents do you have?
--- Min answer (or : w, h, e, d, q, n) : none
ERROR
You must answer the questions correctly
Answer with a number, or
d(o'nt know), or
w(hy), h(ow), or e(xplain), or,
q(uit), or,
na - n(ot) a(pplicable) (score type only)
Press Space bar to continue

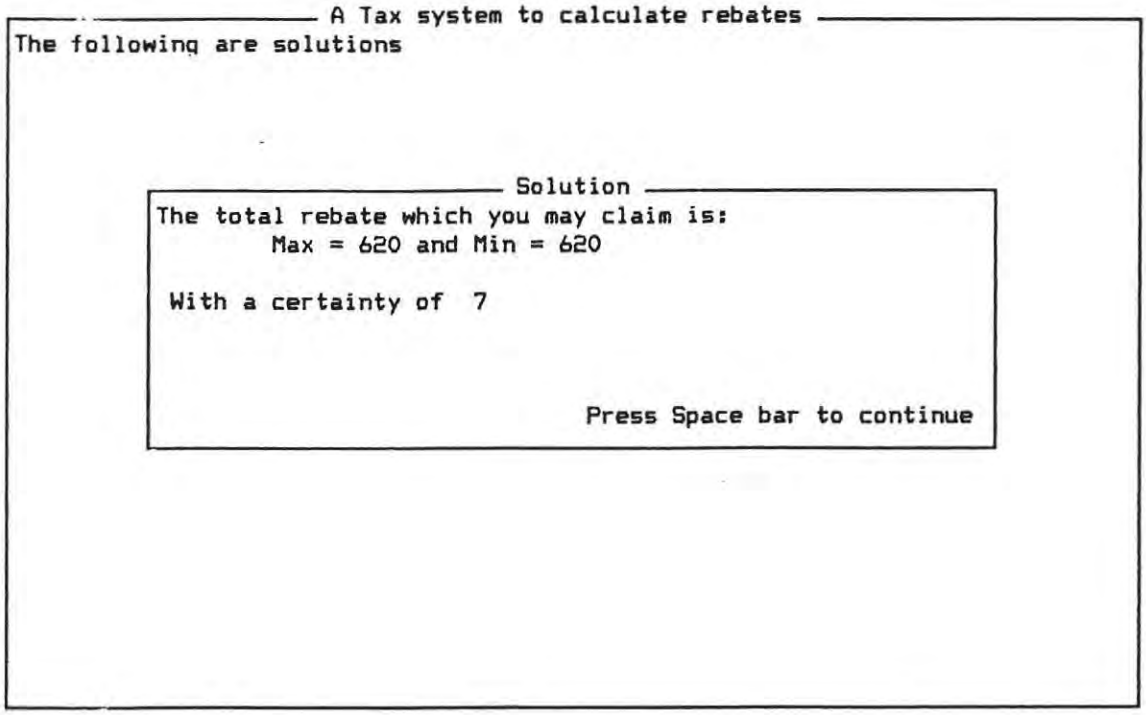

A Tax system to calculate rebates
Calculating the total tax rebate
Calculating the dependent rebate
In terms of the act how many dependents do you have?
--- Min answer (or : w, h, e, d, q, n) : 5
--- Max answer (or: w, h, e, d, q, a) : 3
ERROR
You must make your high answer greater than or equal to your low
Press Space bar to continue

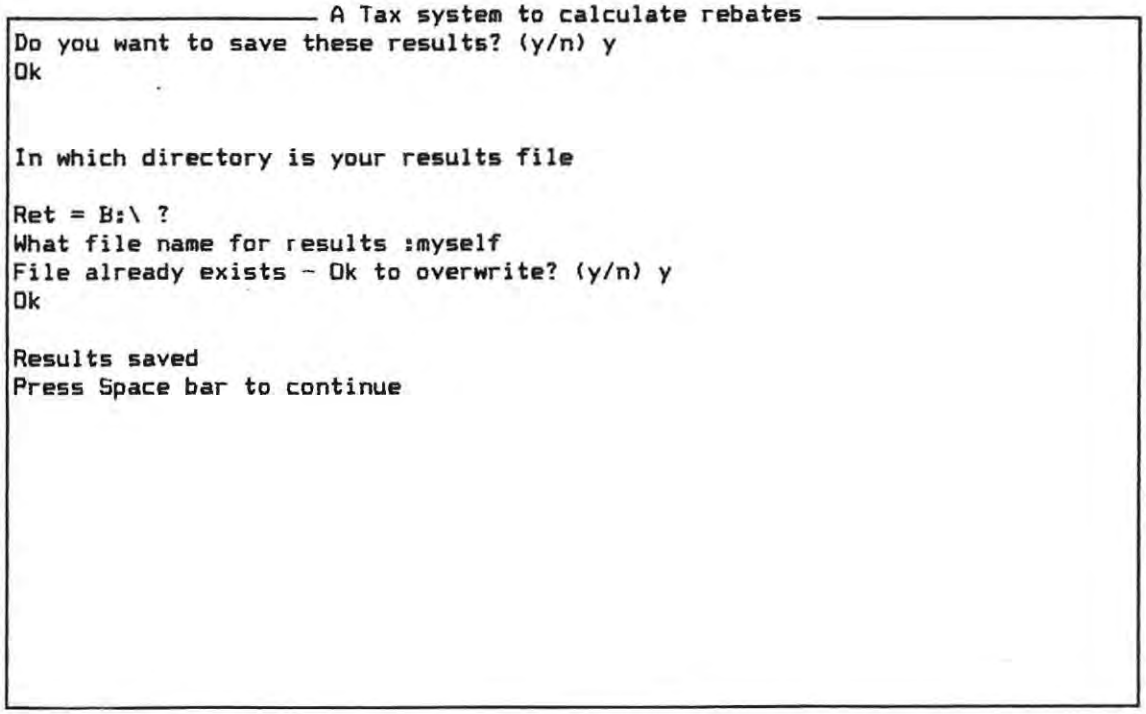

A Tax system to calculate rebates
Calculating the total tax rebate
Calculating the dependent rebate
In terms of the act how many dependents do you have?
--- Min answer (or : w, h, e, d, q, n) : 0
--- Max answer (or: w, h, e, d, q, a) : 0

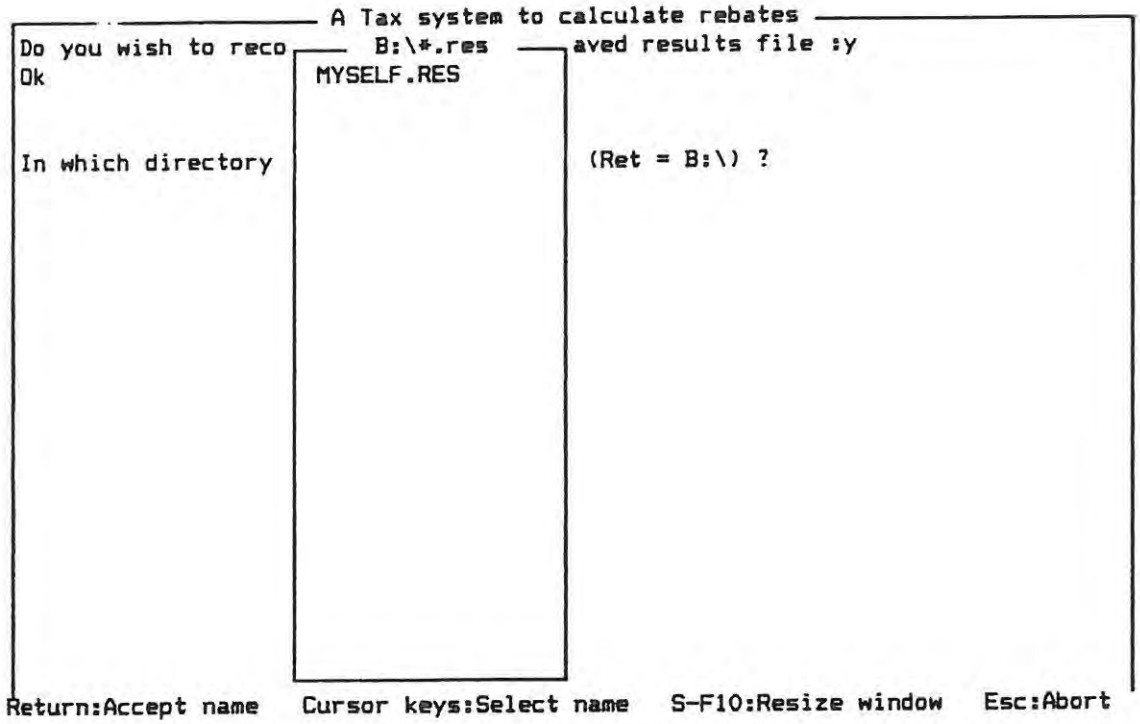

A Tax system to calculate rebates
Calculating the total tax rebate
How much primary rebate can you claim?
--- Min answer (or : w, h, e, d, q, n) : d
--- Max answer (or: w, h, e, d, q, a) : d

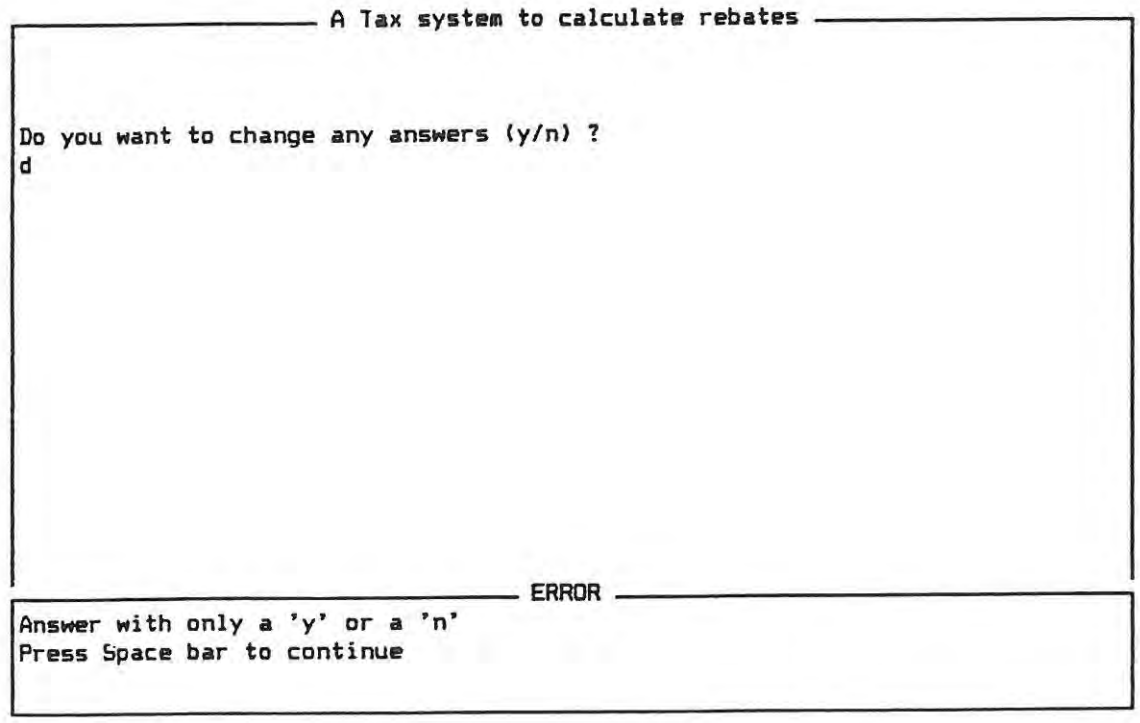

A Tax system to calculate rebates
Calculating the total tax rebate
How much primary rebate can you claim?
Calculating the primary rebate
Are you married in terms of the act?
Answer with y, n or d (or : w, h, e, q) : h
How
If the person is a married person in terms of the act they will be allowed a rebate of R880 otherwise the receiver will allow them a rebate of R620.
Press Space bar to continue

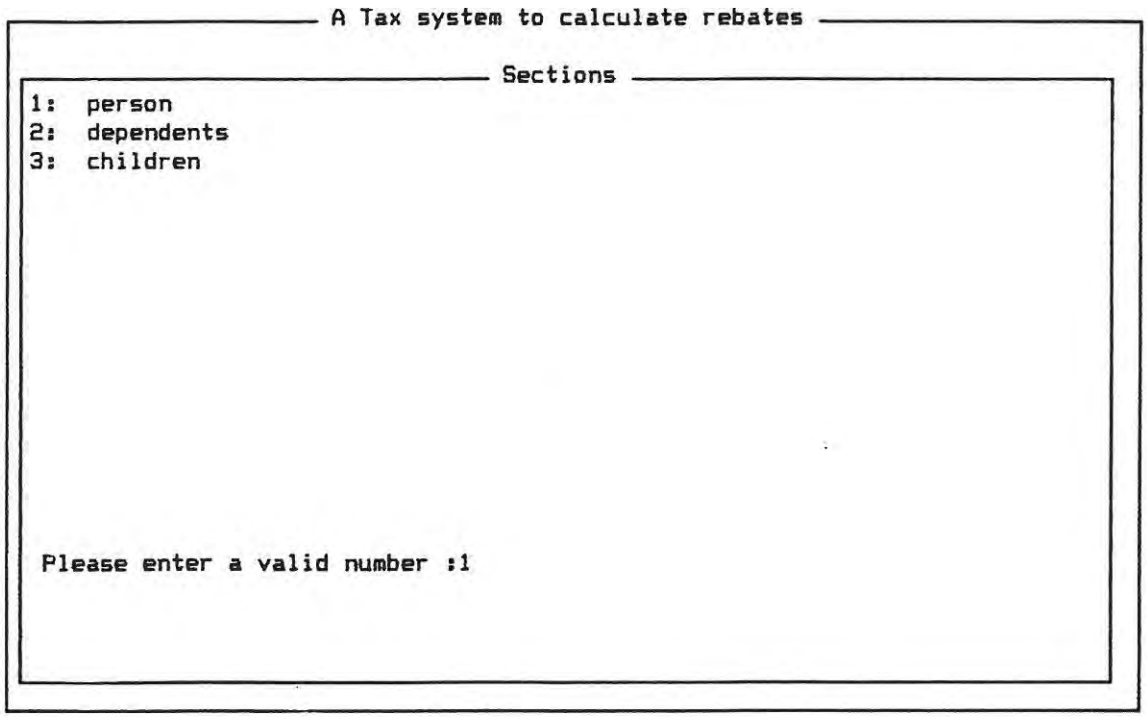

A Tax system to calculate rebates
Calculating the total tax rebate
How much primary rebate can you claim?
Calculating the primary rebate
Are you married in terms of the act?
Answer with y, n or d (or : w, h, e, q) : n
How certain are you of your answer? (1..5, 0=help)0
HELP: Certainties
1 Very uncertain ..
2 fairly uncertain ..
3 certainish ..
4 fairly certain..
5 very certain
Press Space bar to continue

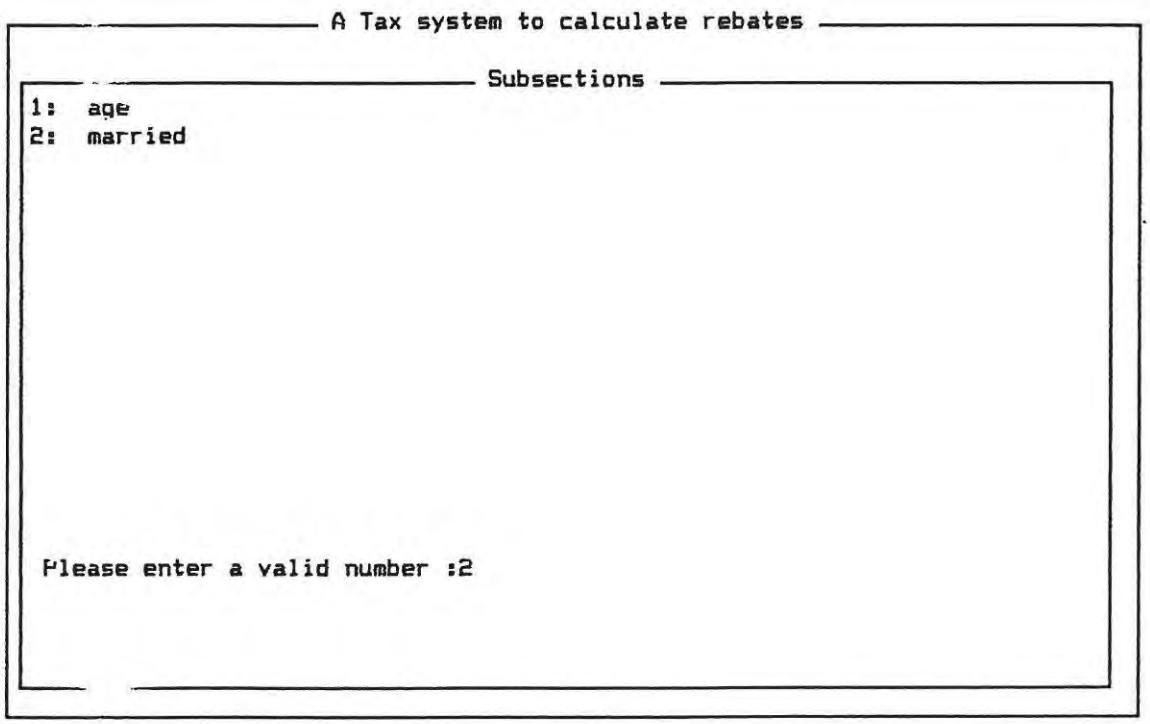

A Tax system to calculate rebates
The following are solutions
Solution
The total rebate which you may claim is:
Max = 620 and Min = 620
With a certainty of 7
Press Space bar to continue

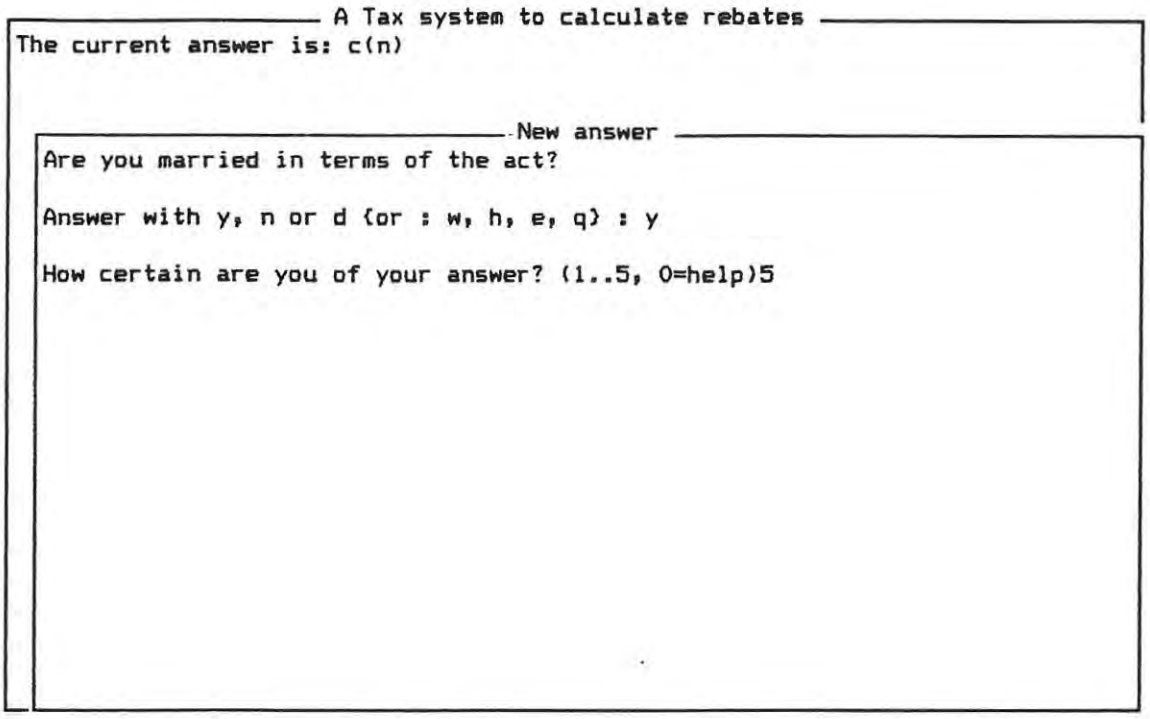

A Tax system to calculate rebates
Do you want to save these results? (y/n) y
Ok
In which directory is your results file
Ret = B:\ ?
What file name for results :myself
File already exists - Ok to overwrite? (y/n) y
Ok
Results saved
Press Space bar to continue

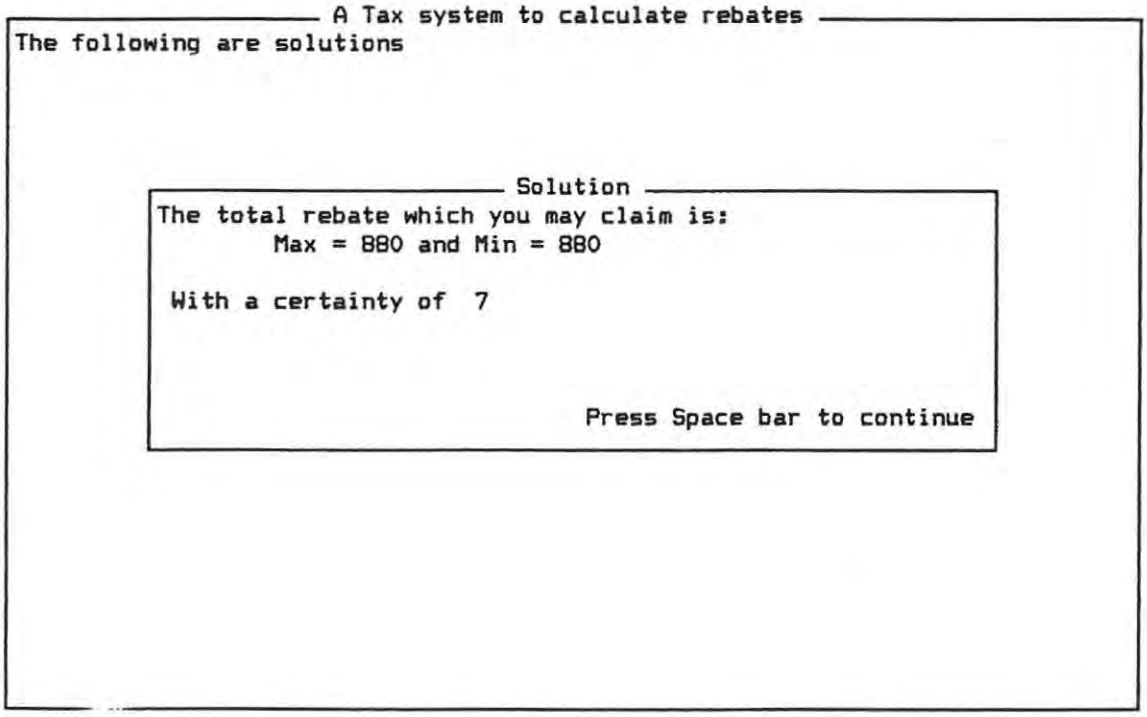

A Tax system to calculate rebates
Do you wish to reco
B:*.res
aved results file :y
Ok
MYSELF.RES
In which directory
(Ret = B:\) ?
Return:Accept name
Cursor keys:Select name
S-F10:Resize window
Esc:Abort


A Tax system to calculate rebates
Do you want to change any answers (y/n) ?
d
ERROR
Answer with only a 'y' or a 'n'
Press Space bar to continue


A Tax system to calculate rebates
Sections
1: person
2: dependents
3: children
Please enter a valid number :1


A Tax system to calculate rebates
Subsections
1: age
2: married
Please enter a valid number :2


A Tax system to calculate rebates
The current answer is: c(n)
New answer
Are you married in terms of the act?
Answer with y, n or d (or : w, h, e, q) : y
How certain are you of your answer? (1..5, 0=help)5


A Tax system to calculate rebates
The following are solutions
Solution
The total rebate which you may claim is:
Max = 880 and Min = 880
With a certainty of 7
Press Space bar to continue

The RCS System

The relative conservation status of this river is :

Min -- 53 and Max -- 74

The percentage score for the river itself is between:

Min -- 60 and Max -- 69

The percentage score for the catchment is between:

Min -- 48 and Max -- 66

The percentage score for the biota is between:

Min -- 44 and Max -- 86

Would you like a more detailed look as to how the results were calculated?

The RCS System

Here is a breakdown for the river section

Subsection	Answers Max	Min	Weight	Weighted answers Max	Min	Research Priority	
mainweirs	0	0	-4	0	0	0	10
sewage	2	0	-16	0	-4	0	10
toxic	9	5	-16	-12	-14	0	10
rubbish	1	0	-6	0	-1	12	8
ecosystems	1	0	7	2	0	14	8
siltation	0	0	-5	0	0	0	10
migration	0	0	5	0	0	0	10

The percentage score for the river itself is between:

Min -- 60 and Max -- 69

Press Space bar to continue

The RCS System

Here is a breakdown for the catchment section

Subsection	Answers Max	Min	Weight	Weighted answers Max	Min	Research Priority	
size	1	0	2	0	0	0	10
vegetation	40	30	18	9	9	18	9
riparianveg	60	40	-12	-9	-9	24	8
forestry	5	2	-6	-1	-1	0	10
arablefarming	50	40	-7	-7	-7	7	9
grazing	20	10	-4	-1	-2	4	9
irrigation	5	0	-8	0	-2	0	10
towns	0	0	-7	0	0	0	10
poplndensity	10	5	-7	-2	-3	0	10
erosion	1	0	-9	0	-2	18	8

Press Space bar to continue

The RCS System

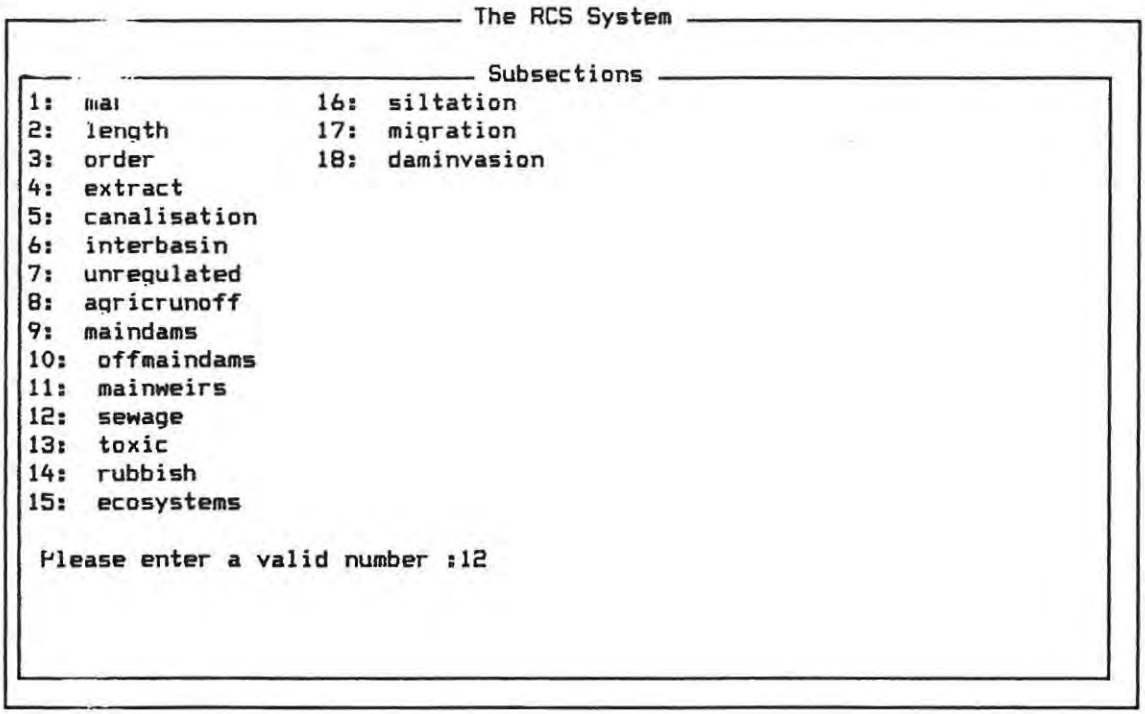
Here is a breakdown for the catchment section

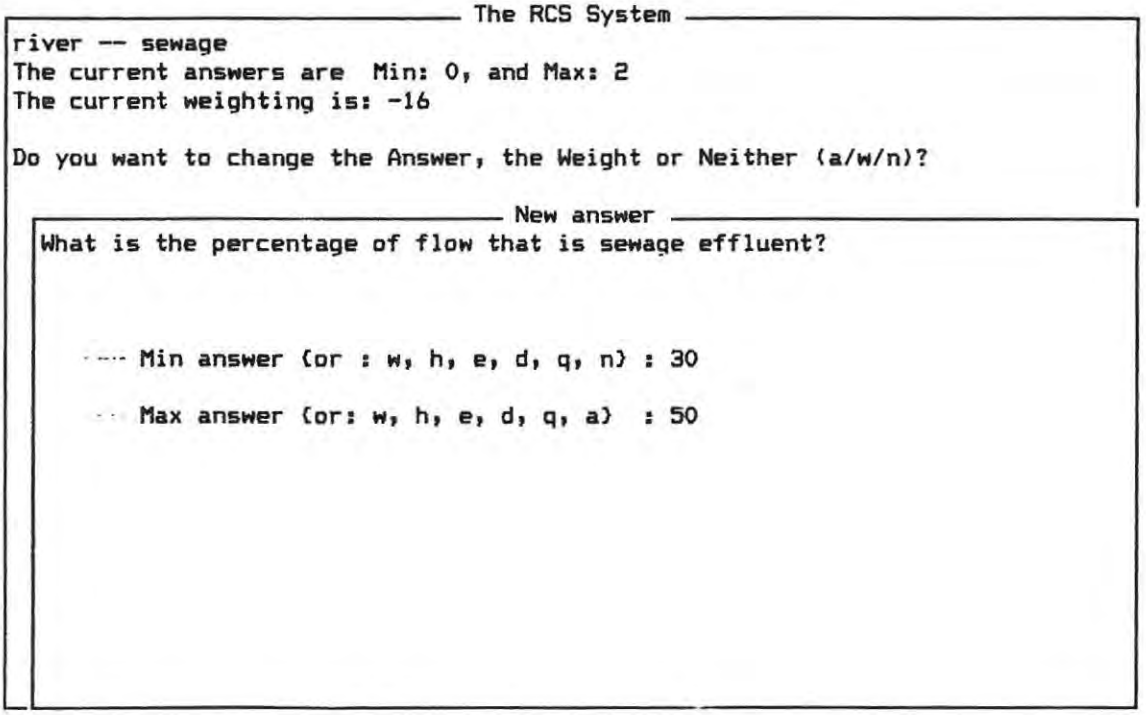
Subsection	Answers Max	Min	Weight	Weighted answers Max	Min	Research Priority	
bankstability	1	0	-5	0	-1	10	8
habitatdiversity	2	1	11	6	3	22	8

The percentage score for the catchment is between:

Min -- 48 and Max -- 66

Press Space bar to continue


The RCS System
Subsections
1: mai
2: length
3: order
4: extract
5: canalisation
6: interbasin
7: unregulated
8: agricrunoff
9: maindams
10: offmaindams
11: mainweirs
12: sewage
13: toxic
14: rubbish
15: ecosystems
16: siltation
17: migration
18: daminvasion
Please enter a valid number :12


The RCS System
river -- sewage
The current answers are Min: 0, and Max: 2
The current weighting is: -16
Do you want to change the Answer, the Weight or Neither (a/w/n)?
New answer
What is the percentage of flow that is sewage effluent?
Min answer (or : w, h, e, d, q, n) : 30
Max answer (or: w, h, e, d, q, a) : 50

The RCS System

The relative conservation status of this river is :

Min -- 48 and Max -- 70

The percentage score for the river itself is between:

Min -- 48 and Max -- 61

The percentage score for the catchment is between:

Min -- 48 and Max -- 66

The percentage score for the biota is between:

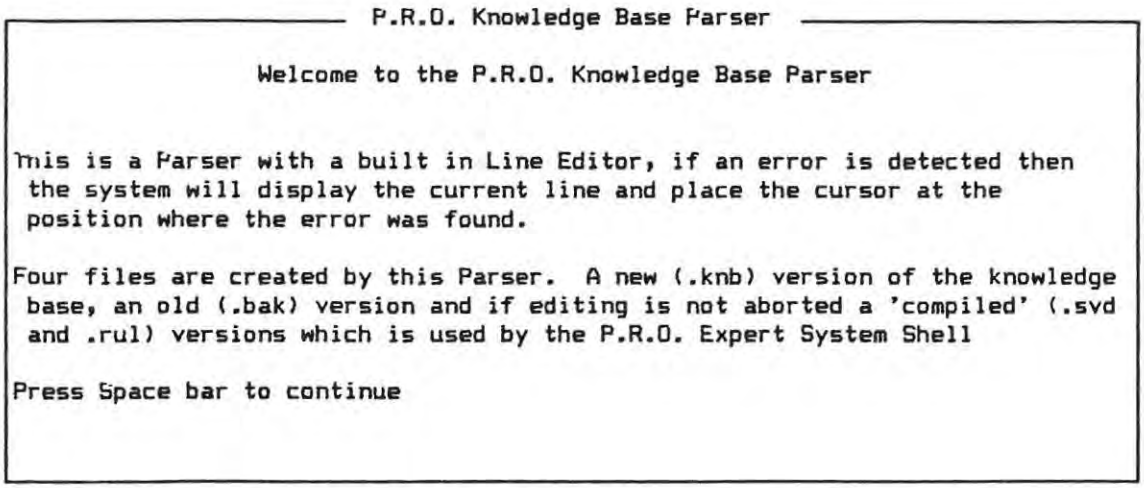
Min -- 44 and Max -- 86

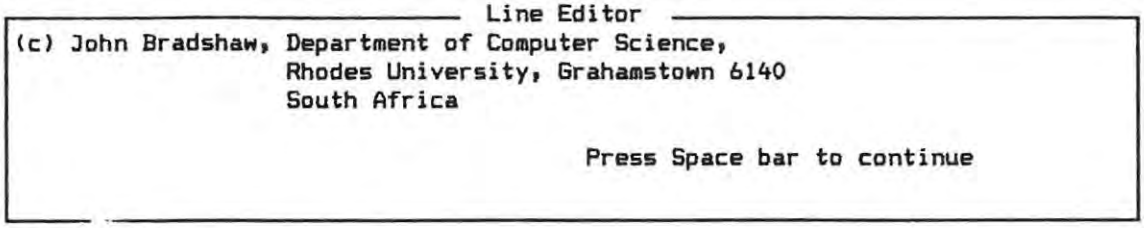
Would you like a more detailed look as to how the results were calculated?

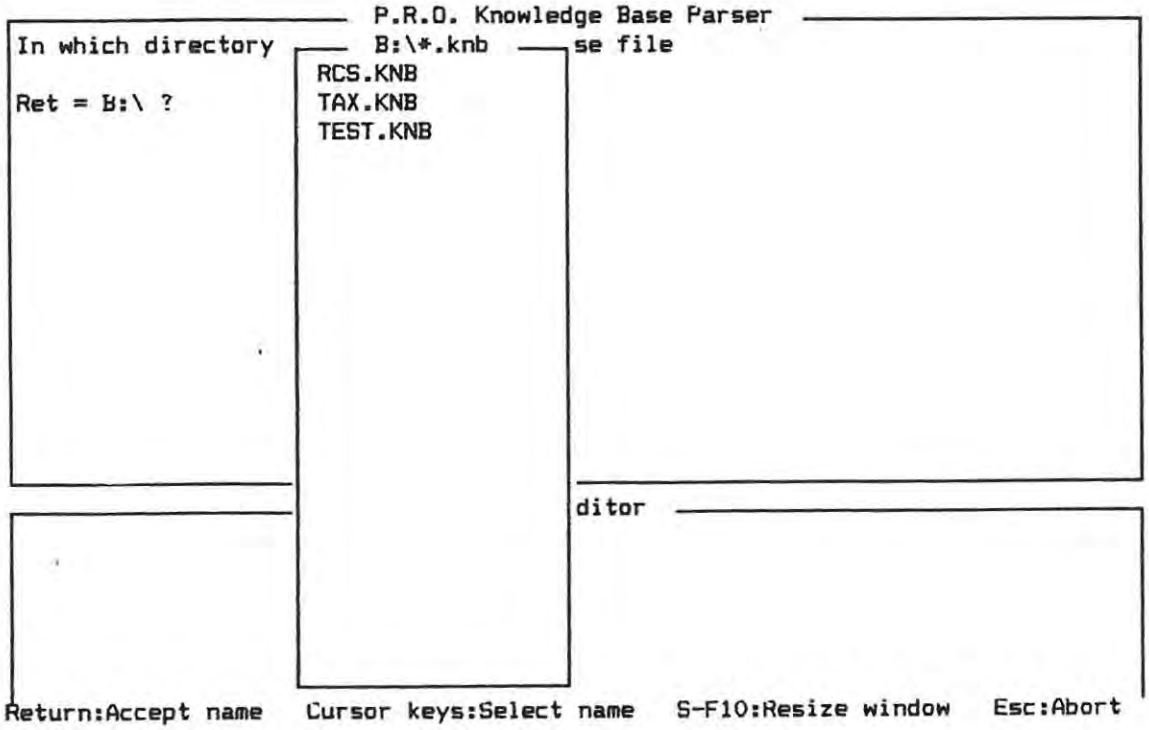
Appendix C

Appendix C contains examples of the P.R.O. Knowledge Base Parser at work.

Again the example tax system has been used. A few errors were introduced into the knowledge base to illustrate the kind of errors recognized by the system. The position of the cursor is not indicated on the diagrams but the column number should reassure the reader that it is in fact placed at the spot where the error is detected.


P.R.O. Knowledge Base Parser
Welcome to the P.R.O. Knowledge Base Parser
This is a Parser with a built in Line Editor, if an error is detected then the system will display the current line and place the cursor at the position where the error was found.
Four files are created by this Parser. A new (.knb) version of the knowledge base, an old (.bak) version and if editing is not aborted a 'compiled' (.svd and .rul) versions which is used by the P.R.O. Expert System Shell
Press Space bar to continue


Line Editor
(c) John Bradshaw, Department of Computer Science,
Rhodes University, Grahamstown 6140
South Africa
Press Space bar to continue


P.R.O. Knowledge Base Parser
In which directory
B:*.knb
se file
Ret = B:\ ?
RCS.KNB
TAX.KNB
TEST.KNB
ditor
Return:Accept name
Cursor keys:Select name
S-F10:Resize window
Esc:Abort

P.R.O. Knowledge Base Parser

```
systemtype(diagnose,
    "A Tax system to calculate rebates",
    [["This is a system designed to calculate tax rebates.  Rebates are",
    "calculated according to four categories - Age, Number of dependents,",
    "the marital status of the person and according to the number of",
    "children the person has.",
    "Answers to the questions should",
    "reflect the status at the end of the tax year. ",
    "This is a very simplified version of the system used in South Africa",
    "for the tax year ending 28 February, 1987."],
    ["This is not an accurate system and the result produced should not be",
    "taken seriously.  This system was written to demonstrate the features",
    "of the P.R.O. Expert System Shell."],
    ["The system was written by John Bradshaw and is based on the TES",
```

Error Correction

```
  Line 1     Col 66    Indent  Insert
    "system designed by Martin Steinhobel at Rhodes University."]).


, or ] expected here
```

P.R.O. Knowledge Base Parser

```
    "for the tax year ending 28 February, 1987."],
    ["This is not an accurate system and the result produced should not be",
    "taken seriously.  This system was written to demonstrate the features",
    "of the P.R.O. Expert System Shell."],
    ["The system was written by John Bradshaw and is based on the TES",
    "system designed by Martin Steinhobel at Rhodes University."]]).

rule(top, rebate, n(H, L)):-
    getvalue(age, rebate, 10, Agehigh, Agelow),
    getvalue(dependent, rebate, 5, Dephigh, Deplow),
    getvalue(primary, rebate, 65, Primhigh, Primlow),
    getvalue(child, rebate, 15, Childhigh, Childlow),
    H = Agehigh + Dephigh + Primhigh + Childhigh,
    L = Agelow + Deplow + Primlow + Childlow.
```

WARNING

```
Your total Ri count should = 100 for each rule
This rule's total = 95

Press Space bar to continue
```

P.R.O. Knowledge Base Parser

```
    "of the P.R.O. Expert System Shell."],
    ["The system was written by John Bradshaw and is based on the TES",
    "system designed by Martin Steinhobel at Rhodes University."]]).

rule(top, rebate, n(H, L)):-
    getvalue(age, rebate, 10, Agehigh, Agelow),
    getvalue(dependent, rebate, 5, Dephigh, Deplow),
    getvalue(primary, rebate, 65, Primhigh, Primlow),
    getvalue(child, rebate, 15, Childhigh, Childlow),
    H = Agehigh + Dephigh + Primhigh + Childhigh,
    L = Agelow + Deplow + Primlow + Childlow.

rule(age, rebate, n(Agehigh, Agelow)):-
    getvalue(over60, rebate, 50, O60high, O60low),
```

Error Correction

```
  Line 1     Col 30    Indent  Insert
    getvalue(over65, rebate, O65high, O65low),


Number ::=  digit {digit}* [. digit {digit}*] expected here
```

P.R.O. Knowledge Base Parser

```
    getvalue(dependent, rebate, 5, Dephigh, Deplow),
    getvalue(primary, rebate, 65, Primhigh, Primlow),
    getvalue(child, rebate, 15, Childhigh, Childlow),
    H = Agehigh + Dephigh + Primhigh + Childhigh,
    L = Agelow + Deplow + Primlow + Childlow.

rule(age, rebate, n(Agehigh, Agelow)):-
    getvalue(over60, rebate, 50, O60high, O60low),
    getvalue(over65, rebate, 50, O65high,
    O65low),
    Agehigh = O60high + O65high,
    Agelow = O60low +O65low.

rule(over60, rebate, n(120, 120)):-
```

Error Correction

```
  Line 1     Col 37    Indent  Insert
    checkvalue(person, age, 100, 1, greater, 60).


eq or lt or gt or leq or geq expected here
```

P.R.O. Knowledge Base Parser

```
    checkvalue(person, age, 100, 1, gt, 65).

rule(over65, rebate, n(380, 0)):-
    resetrule,
    checkvalue(person, age, 100, h, gt, 65).

rule(over65, rebate, n(0, 0)):-
    succeedrule.

rule(dependent, rebate, n(Dephigh, Deplow)):-
    getvalue(dependents, number, 100, High, Low),
    Dephigh = High * 50,
    Deplow = Low * 50 .
```

Error Correction

```
  Line 1     Col 14    Indent  Insert
rule(primary rebate, n(880, 880)):-



, expected here
```

P.R.O. Knowledge Base Parser

```
    Dephigh = High * 50,
    Deplow = Low * 50 .

rule(primary, rebate, n(880, 880)):-
    checkres(person, married, 100, c(y)).

rule(primary, rebate, n(620, 620)):-
    succeedrule.

rule(child, rebate, n(Childhigh, Childlow)):-
    getvalue(children, number, 100, High, Low),
    Childhigh = High * 100,
    Childlow = Low * 100.
```

Error Correction

```
  Line 1     Col 43    Indent  Insert
hypothesis(top, rebate, n, 0, 100, 0,     "Calculating the total tax rebate",



Number ::=  digit (digit)* [. digit (digit)*] expected here
```

P.R.O. Knowledge Base Parser

```
rule(child, rebate, n(Childhigh, Childlow)):-
    getvalue(children, number, 100, High, Low),
    Childhigh = High * 100,
    Childlow = Low * 100.

hypothesis(top, rebate, n, 0, 100, 0, 10,
    "Calculating the total tax rebate",
    ["The total rebate which you may claim is:"],
    [],
    [],
    [],
    []).
```

Error Correction

```
  Line 1     Col 26    Indent  Insert
hypothesis(age, rebate,  0, 100, 0, 10,



c or n expected here
```

P.R.O. Knowledge Base Parser

```
rule(child, rebate, n(Childhigh, Childlow)):-
    getvalue(children, number, 100, High, Low),
    Childhigh = High * 100,
    Childlow = Low * 100.

hypothesis(top, rebate, n, 0, 100, 0, 10,
    "Calculating the total tax rebate",
    ["The total rebate which you may claim is:"],
    [],
    [],
    [],
    []).
```

Error Correction

```
  Line 1     Col 29    Indent  Insert
hypothesis(age, rebate,  n, 0, 100, 0, 10,
```

P.R.O. Knowledge Base Parser

```
hypothesis(top, rebate, n, 0, 100, 0, 10,
    "Calculating the total tax rebate",
    ["The total rebate which you may claim is:"],
    [],
    [],
    [],
    []).

hypothesis(age, rebate,  n, 0, 100, 0, 10,
    "Calculating the age rebate",
    [],
    [],
    [],
```

Error Correction

```
  Line 1     Col 7     Indent  Insert
    []).



, expected here
```

P.R.O. Knowledge Base Parser

```
hypothesis(top, rebate, n, 0, 100, 0, 10,
    "Calculating the total tax rebate",
    ["The total rebate which you may claim is:"],
    [],
    [],
    [],
    []).

hypothesis(age, rebate,  n, 0, 100, 0, 10,
    "Calculating the age rebate",
    [],
    [],
    [],
    [],
```

Error Correction

```
  Line 1     Col 1     Indent  Insert
).



[ expected here
```

P.R.O. Knowledge Base Parser

```
question(person, married, c,
    ["Are you married in terms of the act?"],
    ["The amount of primary rebate and thus the total rebate will differ",
    "if the person is a married person in terms of the act or not."],
    ["If the person is a married person in terms of the act they will be",
    "allowed a rebate of R880 otherwise the receiver will allow them a",
    "rebate of R620."],
    ["The act states that a person is married if he is married in terms",
    "of any law of custom and is living permanently with his spouse, or if",
    "the person is a widow or widower, or if the person qualifies for a",
    "child rebate and is responsible for the maintenance of the child."]).

question(children, number, n,
    ["How many children do you have?"],
```

Line Editor

P.R.O. Knowledge Base Parser

```
    "the year or if the person was under the age of 26 at the end of the",
    "year and was a full time student during the year and was dependent",
    "upon the taxpayer for financial support."]).

weight(primary, rebate, class(620, 620,620,620,620, 880), 0, 0).

weight(person, age, class(0, 0,0,0,0, 130), 0, 0).

weight(dependents, number, class(0, 0,0,0,0, 10), 0, 0).

weight(children, number, class(0, 0,0,0,0, 10), 0, 0).


Press the SPACE bar
```

Line Editor