The University of Aston in Birmingham

METEX : An Expert System for Metamorphic Petrography.

Hugh Dorans

Thesis submitted for the degree of
Doctor of Philosophy

Classification of metamorphic rocks is normally carried out using a poorly defined, subjective classification scheme making this an area in which many undergraduate geologists experience difficulties. An expert system to assist in such classification is presented which is capable of classifying rocks and also giving further details about a particular rock type. A mixed knowledge representation is used with frame, semantic and production rule systems available. Classification in the domain requires that different facets of a rock be classified. To implement this, rocks are represented by 'context' frames with slots representing each facet. Slots are satisfied by calling a pre-defined ruleset to carry out the necessary inference. The inference is handled by an interpreter which uses a dependency graph representation for the propagation of evidence. Uncertainty is handled by the system using a combination of the MYCIN certainty factor system and the Dempster -Shafer range mechanism. This allows for positive and negative reasoning, with rules capable of representing necessity and sufficiency of evidence, whilst also allowing the implementation of an alpha-beta pruning algorithm to guide question selection during inference. The system also utilizes a semantic net type structure to allow the expert to encode simple relationships between terms enabling rules to be written with a sensible level of abstraction.

Using frames to represent rock types where subclassification is possible allows the knowledge base to be built in a modular fashion with subclassification frames only defined once the higher level of classification is functioning. Rulesets can similarly be added in modular fashion with the individual rules being essentially declarative allowing for simple updating and maintenance. The knowledge base so far developed for metamorphic classification serves to demonstrate the performance of the interpreter design whilst also moving some way towards providing a useful assistant to the non-expert metamorphic petrologist. The system demonstrates the possibilities for a fully developed knowledge base to handle the classification of igneous, sedimentary and metamorphic rocks.

The current knowledge base and interpreter have been evaluated by potential users and experts. The results of the evaluation show that the system performs to an acceptable level and should be of use as a tool for both undergraduates and researchers from outside the metamorphic petrography field.

KEYWORDS: Metamorphic petrology, Expert System, Frames, Inference Control, production rules.

To Mum and The Rats!

ACKNOWLEDGEMENTS

I would like to thank Professor D.D. Hawkes and Dr. P. Coxhead for their supervision of this research, and Dr. Coxhead's patience while I attempted to learn some computer science from him. I should also like to thank Brian Drabble for his assistance and encouragement throughout the duration of the project. I should finally like to thank Aston University for financing me during my research and for providing all necessary equipment.

CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Introduction to Problem and Project Aims

In the domain of metamorphic petrography, experts use their personal experience, gathered over a period of years, to reach classifications of rocks on what can be considered a poorly defined, subjective classification system. Currently an international body of domain experts is attempting to simplify the classification through the introduction of an agreed set of guidelines for a systematic approach with accepted definitions for all rock types. Until this is achieved novice geologists working in this area are likely to continue experiencing difficulties in correctly classifying such rocks. It would therefore be valuable to have a computerised classification system which could provide assistance to the novice both by classifying a specimen and by providing a guide as to how the classification should be tackled. Such a system would serve two useful purposes:

1. More straight forward examples could be filtered out so that the expert would only be consulted with more unusual samples.

2. Undergraduates who feel reluctant to approach the expert from a fear of appearing foolish might be more willing to approach a computer system.

Classification by computer has been tackled in many domains and using many different techniques. The most straight forward technique is the use of a decision tree written into the logic of a high level programming language such as FORTRAN or PASCAL. However, such systems are by their nature very difficult to update or alter as the classification is 'hard wired'. This would not be a particularly desirable feature in the domain of metamorphic petrography since the classification scheme has not as yet been systematically defined, so that a system written now will require future alteration. Another approach has been the use of decision tables, often based on coded probability data, or

through comparison of features shown and expected for a particular class. Such a system is much simpler to maintain although a fairly well defined classification scheme would still be required. A more recent approach to automated classification has been through the use of expert systems which allow the knowledge of an expert in some domain of human endeavour to be encoded in a form which can be used by computer systems. The main advantage of such systems lies in the fact that the knowledge can be easily maintained and altered, at least in theory, and in that the systems appear to demonstrate some level of 'intelligence' albeit very limited.

In the past, expert systems have been used in a wide variety of domains including: identifying human diseases, configuring computer systems, identifying mineral prospects and classifying minerals, fossils and igneous rocks. Most of the major systems which have been produced have been research oriented, normally examining some form of knowledge representation, and tend not to have found widespread use. However, recently the availability of cost-effective commercial tools has allowed many people to become involved in the development of expert systems with mixed success. Most of the systems which have been implemented have involved classification of an entity to some single category, although in some cases this requires handling the solution of many sub-goals. In the metamorphic domain it is necessary to classify more than one aspect of a rock so that the domain presents its own particular difficulties. Existing systems with complex control architectures might solve the problem but often these systems are either overly complex or are simply unavailable, normally due to prohibitive costs.

Many different methods of handling the control of inference in complex expert systems have been developed. Of these the major successes have been through the use either of:

1. Meta-rules where declarative control rules are used to activate the lower level inference rules. Advanced systems have rules which are capable of making inference about the knowledge itself.

2. Frame based control or mixed rules and frames which allow clear distinction between control information and inference information. Mixed representations are only now becoming common in so called second generation systems.

3. Blackboard systems which allow disparate knowledge sources to combine in reaching a solution to a problem, each knowledge source solving some sub-problem for which it was designed. The knowledge sources are called by a controlling scheduler, scheduling being based on heuristic knowledge in complex systems.

The aim of this project is to produce, or at least demonstrate, an expert system approach which is capable of handling the problems of the domain of metamorphic petrography. This prototype system should be able to tackle the specific problems of multiple aspects in classification and the provision of a tool of use to the novice or undergraduate geologist. The project system must also solve the problem that metamorphic classification is not as yet properly set out in a systematic form so that the system's definitions used in the classification must be easily updated to meet the new guidelines when available. Once this has been achieved the ultimate aim would be to extend the system to the other main rock types, i.e. igneous and sedimentary, although this is not a direct aim of this project.

The system presented in this thesis represents a solution to the metamorphic classification problem. The system is able to classify the main aspects of a metamorphic rock whilst also providing the user with additional general details about rock types, the details being provided in advance by the expert. The representation is a mixed frame and rule representation in which rulesets are partitioned into knowledge sources which assist in the solution of the problem as requested the scheduling control system. In this way the control in many ways resembles a simple blackboard system where scheduling of problem solutions is carried out in advance by the expert. In addition the system has limited scope

to alter scheduling in special cases which are normally foreseen in advance by the expert who must write specific re-scheduling rules. Upgrading the system is possible since the knowledge base is built in a modular manner with new classifications being bolted on as required at the appropriate point in the search space. This includes the ability to reduce the current search space to a sub-section of a larger search space by adding a higher level control units.

1.2 Thesis Structure

CHAPTER 1 - Gives a brief introduction to the thesis detailing the aims of the thesis and its structure.

CHAPTER 2 - Reviews the problem of classification with particular reference to metamorphic petrography. These problems then lead on to a review of Intelligent Knowledge Based Systems (IKBS) which have relevance in the design of the interpreter built for this project.

CHAPTER 3 - Amplifies the important problems specific to metamorphic classification. Having restated these problems the different approaches to tackling them are briefly considered along with the different programming languages and environments which might be applied.

CHAPTER 4 - Discusses in detail the design of the inference system, used to apply rules specific to some small subset of the overall classification. This includes details of the hybrid uncertainty handling system developed for use in inference.

CHAPTER 5 - Discusses the design of the overall control architecture of the interpreter. This includes details on the passing of control between different knowledge structures and sources, the use of procedural attachment and question definitions to interrogate the user and the use of semantic

information in achieving meaningful rules. This is accompanied by a discussion of the considerations which are important in controlling backtracking.

CHAPTER 6 - Details the features developed in the knowledge base of METEX. The individual tasks in metamorphic classification are considered in turn, giving the subclassifications available in each along with details on areas which are particularly problematic or contentious.

CHAPTER 7 - Presents an evaluation of the METEX system, considering the applicability of the approach taken to provide a computerised classification for metamorphic petrography.

CHAPTER 8 - Presents a discussion of further work based upon this research.along with conclusions about the work in this thesis.

APPENDICES:

APPENDIX 1 - A brief user guide, detailing the use of the METEX system as it exists on the Aston University VAX Cluster. This guide should provide new users with enough information to get started with the system.

APPENDIX 2 - Detailed information for prospective knowledge engineers. The information should allow maintenance and updating of existing knowledge bases or creation of new ones.

APPENDIX 3 - A brief summary of the major variable and functions available for knowledge base building along with their correct usage. This is followed by a summary of the syntax of the main knowledge structures.

CHAPTER 2

Review of Related Literature

2.1 Classification Schemes in Geology

Classification in geology, as in many other domains, is used as a tool through which the geologist is able to make clear comparisons between similar rock specimens. Successful classification requires the use of a set of clearly recognizable and meaningful features which may be easily divided into distinct categories. The categories used should ideally have clearly defined boundaries which allow no subjective judgement in choosing how the feature is to be categorized. Combinations of these features are then used in forming the basis of a set of 'pigeon-holes' into which an entity, in this case a rock , may be placed.

The sub-domains of geology which involve classification have shown varying degrees of success in producing clearly defined working schemes. In palaeontology, for example, the classification scheme used is similar to that operated by biologists and, as a result, is very well defined in most instances. This is mainly due to the availability of natural breaks on which to base categories. Mineral classification shows a similar clear definition when data from all of the available tests is used. However, as fewer tests are used and mineral grains become smaller the amount of subjective judgement, in many instances, increases. In the domain of rock classification the definition of the classification schemes used is on the whole reduced, with subjective judgement being increasingly used for certain of the major rock divisions. This reflects the fact that in many cases the categorization of rock features is based on a set of arbitrary, and often meaningless, boundary conditions. This potentially leads to arguments about the nomenclature of rocks, due to overlaps occurring, whenever more than one feature is being considered.

2.1.1 General Aspects of Rock Classification

Petrological classification is based on a wide variety of physical features shown by an individual rock specimen. The main criteria which are used are those of composition and fabric, both of which have genetic importance to the geologist. These two main features are further subdivided into a group of important features which can be quantified with little difficulty if necessary. Compositional features of a rock are divided into the mineral percentage, or modal composition, and the chemical composition. The former is a quantified percentage distribution of the minerals present in the rock. The modal composition tends to be the main feature used in classifying igneous rocks and most textbooks, such as Hatch, Wells and Wells (1972) and Best (1982) carry classification tables based upon modal compositions. The chemical composition of a rock is represented by a quantified percentage distribution of the element oxides and trace elements present in the rock. Chemical compositions are widely used by geochemists to distinguish between the environments in which igneous rocks have been formed. Good examples of this are the Pierce and Cann discrimination plots for basalts (Pierce and Cann 1973). The general term 'rock fabric' also covers two main areas, often referred to as fabric and texture. The texture of a rock is a description of the features and interrelationships of individual grains on a microscopic scale, while the fabric represents a description of the macroscopic features and inter-relationships of grain aggregates. Rock fabrics are used normally to further subclassify rocks of a particular composition. For example, grain size, which is a textural feature, is used in rocks of granitic composition to distinguish between, granite, micro-granite and rhyolite. Textural information is of further use in indicating the history of a rock and detailed studies in sediments can reveal important information on diagenetic history, while studies of metamorphic textures can reveal details of metamorphic processes (Spry 1969).

Unfortunately, in terms of classification, all of these features show continuous variation across the rock spectrum with few natural breaks. In order to produce classifications, geologists have been required to set up arbitrary break points for

categorizing these features. Many of the categories set up in this way, by petrologists, have direct genetic meaning; others however have no such meaning, leading to a great deal of confusion where workers disagree as to the boundary conditions. A good example of this type of arbitrary boundary might be a quartzite which is defined as a rock which is essentially monomineralic and composed predominantly of quartz. The problem here is in choosing a percentage of quartz (e.g. 70, 80 or 95%) which delimits the category of monomineralic. There is, in fact, no natural 'break point' so that one geologist will have some subjective 'break point' which may or may not differ from that used by another geologist. Coupled to this problem is one of nomenclature where either:

1. More than one name is possible depending on what classification feature has been used. For example Hatch *et al.* (1972, pp306-308) describe the methods for distinguishing between the dioritic and gabbroic groups of igneous rocks. Petrologists are able to distinguish between these rock groups based upon either plagioclase composition, the type of mafic minerals, quartz content or colour index. Unfortunately it is almost impossible to reconcile all of these features in any one specimen, producing a situation where different geologists may attach a different name to the same rock.

or

2. More than one synonymous name exists for the rock so that individuals may use different names in different countries. For example in the UK a geologist might classify an igneous dyke rock as a dolerite but in America the term diabase would be used.

In rock classification, one of the first decisions to be made is into which of the major divisions (igneous, sedimentary or metamorphic) the specimen under consideration is to be placed. It should be noted here that in most cases this is not a difficult task although some rocks do present problems, particularly where the hand specimen or thin section is all that is available. A good example of the type of problem which can occur might be

found in the case of a rock which in hand specimen is crystalline with a composition of almost 100% calcite. Is this a limestone (sedimentary), a carbonatite (igneous) or marble (metamorphic)? Fortunately situations where specimens are available without some knowledge of the field associations are rare, although they do occasionally occur in museums and in laboratory teaching collections.

Assuming that a specimen can be assigned to one of the three major rock divisions, further classification then becomes a task of working through the systematic schemes available for the division in question. In the case of igneous or sedimentary rocks this can be, in many ways, a fairly straight forward task due to the availability of agreed systematic schemes of classification, as provided in general texts such as Hatch *et al.* (1972), Best (1982), Jackson (1970) and Greensmith (1978). However, problems do occur even in these well established divisions, as shown by the diorite/gabbro problem discussed above. These difficulties probably arise due to the classifications being strongly based on compositional evidence, particularly in the igneous rocks. Since, as previously stated, the categories of composition used are in many instances arbitrary, it is hardly surprising that problems occur. This results in a certain amount of disagreement between experts as to exactly what nomenclature should be used, resulting in borderline cases being judged through personal experience on the part of the expert. In addition there is often an inherent inaccuracy in the collection of the compositional data. Although geochemical data tends to be generated with relative accuracy, by analytical instrumentation such as X-Ray Fluorescence (XRF) and Electron Probe Micro Analysis (EPMA), modal composition tends to be either visually estimated or statistically calculated from point counts or area data. Both point counting and estimation techniques are accepted as having a biased weighting toward the darker minerals in a specimen so that both methods produce inaccurate results, often compounded by the small samples used. Despite such problems the classifications do tend to work, without much difficulty, in the majority of cases. Only the precise definitions of borderline specimens appears to present major problems.

2.1.1.1 Metamorphic Classification

Although working classifications do exist the task of classifying metamorphic rocks tends to be subjective. This is because no clearly defined, systematic, classification scheme has, as yet, been agreed upon by the experts. The major problem with metamorphic classification is that the rocks show a variety of features useful for classification, each having different but clear genetic meaning. However, these features have differing degrees of importance to individual workers. While one worker might be interested in a compositional differentiation of the rocks, another is interested in a differentiation based on the metamorphic fabric. Fortunately it is possible to combine these two schemes to suit both workers. This is generally done by using a fabric based name where one is clearly indicated. Prefixing this name with a list of the important minerals then gives an indication of the important compositional characteristics, for example 'quartz mica schist'. However, if the rock can be clearly placed in a compositional class, then the name given should be based upon this class, possibly prefixed with a suitable term to describe the important features of the fabric. Although this sort of scheme works well for cases where the rock falls into one clearly defined class, problems do arise when the rock shows features of many different classes and could equally well belong to any of them. For example, imagine a specimen which meets the following description:

> 'Medium grained with about 70% quartz, 10% feldspar, ~5% chlorite and 10 to 15% biotite. The chlorite and micas are segregated into bands which show some schistose fabric.'

Some geologists might name this rock as a quartzite, others however might use the name gneiss in a purely textural terminology, yet another geologist might use the term 'banded schist'. All three of the above names would be equally applicable in the absence of any hard and fast set of rules defining the systematic classification of metamorphic rocks. It would possibly be of more value to make use of an intimate combination of terms and name the rock either, a 'gneissose quartzite', or, more explicitly, a 'schistose banded quartzite'. Although, as the name becomes more cumbersome it may also become less

useful, and if these terms were set up as distinct categories, then we would have to have a set of subjective boundary conditions for the categories.

The complexity in applying terms to metamorphic rocks, as just described, must be accepted in view of the very nature of the rocks themselves. Metamorphic rocks, by the definition of their name, are produced by alteration, or 'metamorphism', of some previous rock type. This metamorphism can occur under a wide spectrum of conditions of pressure (P), temperature (T) and volatile fluid composition (X), described as the metamorphic facies. These widely variable conditions produce an equally wide spectrum of changes in the precursor rocks, which, we should remember, could have been from any of the three major rock divisions, i.e. igneous, sedimentary or metamorphic. This means that while some workers might be interested in what rock type has been metamorphosed and thus use a classification which caters for their needs, using terms like 'meta-granite' or 'meta-pelite', other workers might be interested in the metamorphic history of the rock and so be more concerned with a facies classification for the rock, based on categorization of particular features caused by the varying P, T and X conditions. This variety of interests in the metamorphic rocks therefore requires that, rather than use one single classification based upon personal requirements, some combination should be used, in order to convey a complete picture to other workers. However the most useful terminology, in most cases, is the general classification, and it is this classification which is used by most petrologists to describe a rock specimen.

General Metamorphic Classification

As described above, metamorphic rocks are normally classified into either a compositional class or a textural class depending upon which is applicable. Although most researchers do agree on the definitions to be used for various terms and classes, there is no officially agreed system of classification. However, some progress is being made towards this goal through the SubCommision on the Systematics of Metamorphic Rocks (SCMR),

on behalf of the International Union of Geological Sciences. At the moment this initiative to form an organized standard has progressed only to the stages of requesting first comments on the proposed definitions. Hopefully the results of this first stage in the proceedings will soon become available, thus clarifying just what the majority of metamorphic petrologists envisage as forming the correct method of classification. In most instances there is little or no problem in using the existing terminology, although some confusion over the definition of terms does exist, something which should hopefully be ironed out by SCMR. The main difficulty arises where terms have been used incorrectly in the literature for a period of time and have become so deeply rooted that it would be very difficult to abandon their use. A classic example here is the term 'granulite' which is sometimes used as a general classification term for rocks of granular texture, among the compositional/textural classes such as schist or quartzite. However, the term 'granulite' should strictly be reserved for those rocks which clearly belong to the regional hypersthene grade (Winkler 1979), or granulite facies of metamorphism, and is not therefore a suitable term for use in general classification. Care should therefore be taken as to the terms which are used, since if they are out of context, as in this case, but begin to be widely applied, then extreme difficulties can occur when an attempt is made to clarify the definition of the term. Such a difficulty is well expressed by Winkler who makes an unsuccessful attempt to resolve the problems with the term 'granulite'. Winkler proposed that the term 'granolite' be used as a fabric based term in place of 'granulite' where the hypersthene grade was not shown, but granoblastic textures were present. However, such a scheme has unfortunately never been widely accepted since the use of the 'granulite' term is so deeply rooted.

Precursor Classification

In many examples of metamorphic rocks it is possible to give a rough guide to the original nature of the rock before metamorphism took place, a feature important to deciding upon the minerals which should be used in facies classification (Jackson 1970),

(Fry 1984). This is normally done by using a combination of compositional, textural and field relationship data. Field relations are a useful tool, where available, since particularly in the higher grades of metamorphism, much, if not all, of the original texture is destroyed and compositional evidence can easily become ambiguous in the absence of textural information. In these cases, it is only the field relations of the rock which are likely to indicate whether the original nature was igneous or sedimentary. Definite classification of a precursor type is only truly possible in cases where the compositional and relict textural data suggest a rock type with which the field relations correlate. For example if a rock were to show a mafic composition, with no other information, we could assume that it either resulted from metamorphism of a mafic rock or from a calcic mudstone, as either could yield a mafic composition. If however the rock shows some relict ophitic texture then it has clearly resulted from metamorphism of a mafic igneous precursor. However, without the field relation data for the rock, it becomes difficult to show whether the specimen was once a mafic pluton or a much smaller dyke type intrusive. It should be noted here that this is not a specific problem for the metamorphic petrologists since even the exact classification and understanding of a fresh igneous or sedimentary rock requires the use of field relationships. However, in the case of metamorphic rocks the problems are multiplied. Since all of the related rocks have been altered by metamorphism it is often required that the complete suite be analysed together before a complete interpretation can be made. Also, in the case of regional metamorphism the alteration of the rocks is accompanied by pervasive deformation, in the form of folding and shearing, often making the task of discovering the pre-metamorphic relationships of rock bodies a very difficult one.

Facies Classification

As described above metamorphic studies revolve around a set of rocks which, by definition, have undergone change under specific conditions of pressure and temperature. Through the detailed study of the mineral phases stable at different pressure/temperature

conditions, and different bulk rock compositions, along with a clear understanding of the thermodynamics of solid state chemistry which affects the modal composition of metamorphic rocks, metamorphic petrologists have found ways of estimating the P/T conditions under which a particular rock has been metamorphosed. This is done by looking for the stable mineral assemblages in a specimen and comparing these to the assemblages reported in the literature. This technique produces a facies classification for the rock, such as is shown in Figure 2.1, from which it is possible to convey P and T conditions for comparison of one specimen with another. As with the basic rock classification, no officially recognized standard has been agreed for this facies classification. Although most researchers use similar mineral assemblages for their categories, several different schemes exist, for example Jackson (1970) shows a comparison between the schemes defined by Turner (1968) and Hietenan (1967) against the more general zone concept of metamorphic grade. More accurate calculations of the pressure and temperature of metamorphism is possible if particular mineral species are found in thermodynamic equilibrium. By making use of the compositions of these minerals, it is possible to calculate the thermodynamic conditions directly. This technique, however, requires detailed chemical analysis by Electron Microprobe and also suffers from a lack of agreement on the methods to be used in calculating the P and T. Minerals used in this way are termed geo-barometers and geo-thermometers. The approach of quoting approximate P and T conditions is slowly becoming more popular as workers move away from the ambiguities of facies categories. However, facies, and subfacies, categories are still widely used by most geologists.

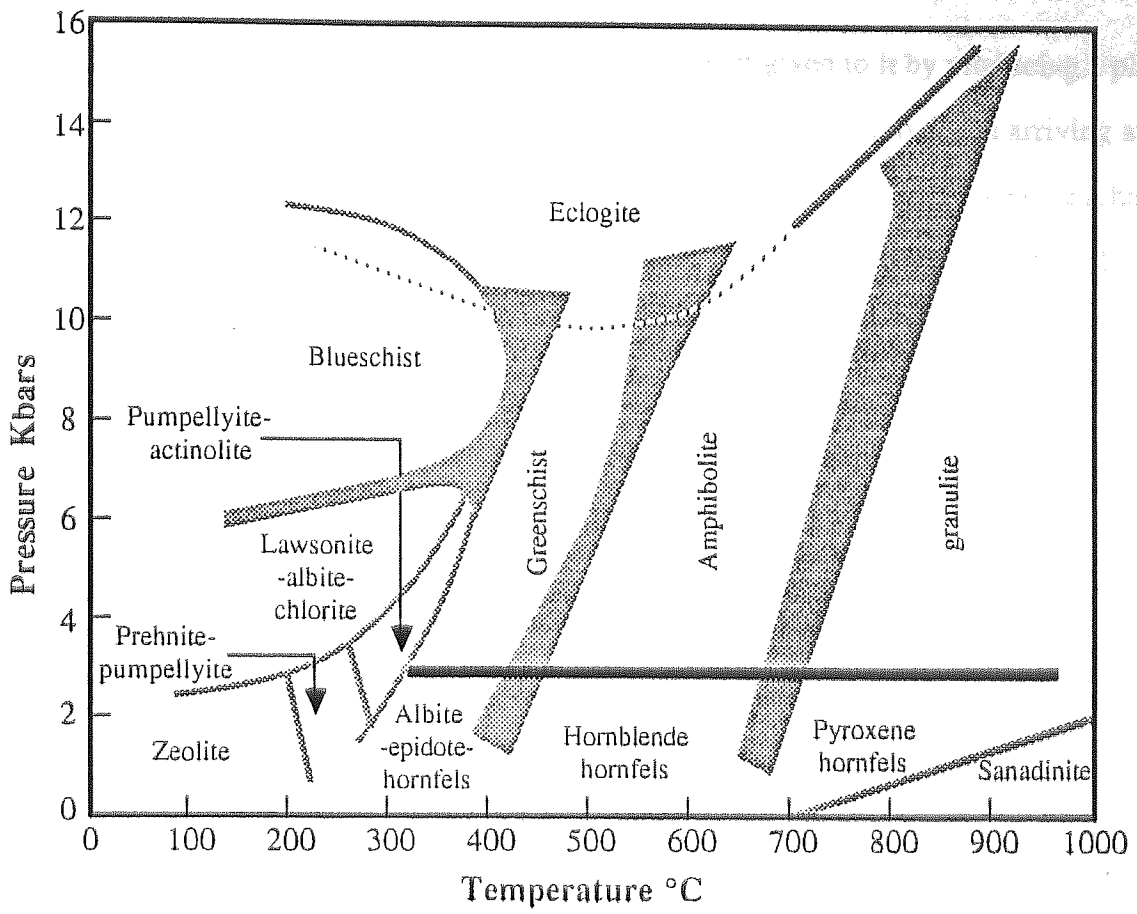

Figure 2.1 Schematic diagram showing relationship of common facies terms to pressure and temperature. (After Turner 1981)

2.2 Brief Introduction to IKBS

Intelligent Knowledge Based Systems (IKBS), and more specifically expert systems, are a special group of problem solving programs developed by Artificial Intelligence (AI) researchers. AI research in the late 1960s was concerned with general problem solving programmes in an attempt to model human abilities in performing simple reasoning tasks, an approach which stemmed from an early belief that the power of human reasoning came from a set of powerful and general reasoning techniques. In general the programs developed at this time were limited to solving what were essentially toy problems as opposed to solving real world problems. Many successes were reported from this early research, although many workers now believe that much of the success was due to fine tuning for specific problems. The General Problem Solver (GPS) (Newell &

Simon 1972) was intended to be able to solve any problem given to it by producing a plan type solution. To do this GPS required to be given a set of operators to use in arriving at a solution, a set of descriptive statements about its world and a goal to be reached. Unfortunately the criticism levelled at GPS is based on the need for a full problem description and complete set of operators to be given to the system. By the time this data is made available the solution becomes almost obvious. Despite this GPS was successful on the toy problems on which it was developed but never performed adequately on the real world problems for which it was originally conceived.

Research prior to the late 1960s had been strongly centred around exhaustive search algorithms. This research on searching algorithms has found extensive use in the games and puzzles which were thought to require intelligence to perform well at, and is in fact still a major area of research. In the more complex games such as chess the successes using searching for problem solving were, however, only achieved after the development of heuristic search techniques which could limit the area of the search space being considered. For a full discussion of search techniques in AI the reader is referred to any of the introductory texts (e.g. Bundy 1980, Hunt 1975, Rich 1983, Winston 1984), or for a fuller discussion of heuristic search see Pearl (1984). Heuristic search involves the use of 'rule of thumb' type information in choosing the best path to use in searching for a solution, that is the paths which, with a limited amount of information, look most likely to lead to a solution. For example, in a game of chess the system requires to choose a move which may be valuable, in leading to a win, without requiring to evaluate every possible move to the end of the game. According to Winston (1977) a conservative estimate of the potential position nodes in an average game tree might be about $2.5 * 10^{154}$ assuming a consistent branching factor, from any position, of 35 with 100 moves in the game, some estimates run to much larger figures. Since the computation of so many game states is realistically impossible it is necessary to limit the searching (evaluation) to a small subset of the branches which are possible and to a restricted number of moves ahead. Although the search routines developed for this task cannot be regarded as expert they do form one

of the earliest stages at which 'domain specific knowledge' was introduced to systems in order to assist problem solving.

At this stage researchers became aware that human reasoning abilities in general problem solving could not be modelled by search algorithms alone. In fact, it was realized that humans are able to perform reasoning tasks over a wide variety of problems, not necessarily because of powerful reasoning techniques, but rather because of the vast amount of general knowledge which can be brought to bear on a problem. It would obviously be a massive task to attempt to encode such general knowledge into a computer system, and would probably prove impossible with the current technology. Due to this problem it is now accepted that intelligent, human-like performance can only be successfully modelled for very narrow and specialized areas, or domains, of human endeavour. This has led to the development of so-called expert systems, which attempt to model the problem solving performance of human specialists in particular domains of knowledge. In this context an expert is someone who applies very specialized knowledge to solving problems in a very narrow domain, the knowledge used often having been learned in the later stages of education and augmented by vast experiential knowledge. The extraction and encoding of this knowledge is termed knowledge engineering, as described by Fiegenbaum (1977, 1979). The first such expert system was developed at Stanford for use in analysing chemical spectra data to work out the likely structure of complex organic molecules. The system was called DENDRAL (Fiegenbaum *et al.* 1971). DENDRAL operated in a domain which is very expert in nature and has been reported as performing very well in its domain although, as with many of the early 'successful' systems it does not appear to be used as widely as it might.

Since this time many more expert systems have been developed in a wide variety of domains, some performing with considerable success: R1 (McDermot 1982a, 1982b) for configuration of computer hardware; MYCIN (Shortliffe 1976) for clinical diagnosis and treatment; PROSPECTOR (Duda *et al.* 1978) for interpretation of geological data with respect to different models for mineralization; others with less success, although the

literature only tends to cover the successes. The variety of systems called expert systems which have been produced in the past decade are not all in fact expert, either in terms of their performance or the domain in which they operate, although they do use encoded human knowledge. A better term has therefore been introduced to categorize such systems without referencing their performance, that of Intelligent Knowledge Based Systems (IKBS).

2.2.1 General Architecture of IKBS

There is a great deal of discussion within the IKBS community currently, regarding the software engineering principles which might be applied to IKBS design. As yet no clear principles have emerged except general outlines on what modular units ought to be provided within the overall system. It is generally agreed that IKBS are similar to more conventional computer systems in that where a conventional system = program + data, an IKBS = interpreter + knowledge. Following this 'equation' it has become the convention to regard the interpreter, which uses the expert knowledge, and the knowledge itself as completely separate sections of the system as in Figure 2.2. This approach is regarded as correct since it allows the knowledge to be maintained much more easily since it is clearly identifiable rather than being tangled up within the interpreter code. It should also be theoretically possible to apply the same interpreter to other domains by simply changing the knowledge base, an approach first tested in the EMYCIN system (van Melle 1979, van Melle *et al.* 1984). This is the approach taken in the commercial shell development tools, and has had wide success in developing smaller knowledge bases, although successes in larger knowledge bases has been more limited since, in reality, the interpreter does appear to require some domain specific features. This could be argued however as being a fault of the representations used in IKBSs since they do not allow a completely general purpose interpreter. As more research is carried out in the fields of machine learning and also inference control (see Section 2.2.3) this problem may be solved and more general interpreters produced.

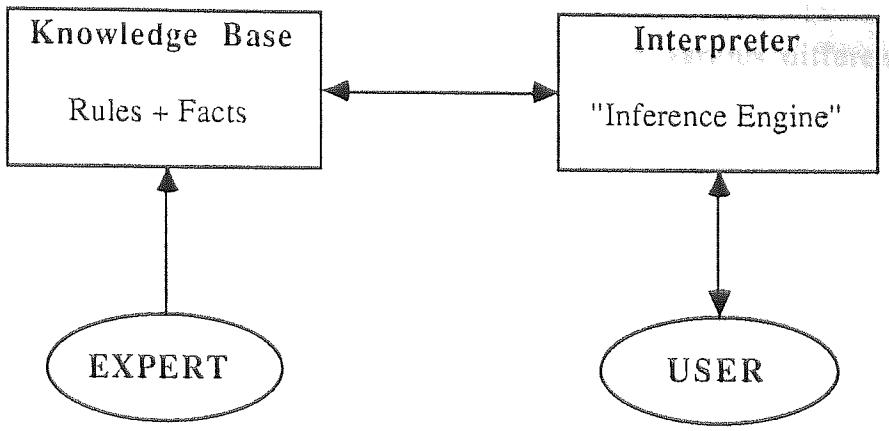

Figure 2.2 Generally accepted ideal architecture for expert system design with separate interpreter and knowledge base.

Within this structural breakdown of IKBS architecture, many workers have laid down a suggested group of facilities which they believe should be available in any IKBS (see Figure 2.3). The system as a whole observes the restriction that the knowledge base and interpreter modules should be separate entities. The interpreter however becomes more complex, since it must be capable of providing explanations as to the reasoning behind a chosen solution. This ability to explain its reasoning is perhaps one of the most important features of any IKBS. Without this ability no human user is likely to feel comfortable with the system's decisions, particularly in areas such as medical diagnosis and financial consultancy. The ability to explain its reasoning is perhaps one of the features which distinguishes an expert system from a more conventional probabilistic classification system which could certainly not be stopped and asked to explain its reasoning (Davis 1987). The Man Machine Interface (MMI) used in an IKBS tends to be relatively simple in that the questions and explanations are formed by a pattern matcher and canned text. More sophisticated interfaces are being provided however, which attempt to simulate natural language interactions with the user and the expert. Although natural language interaction is a desirable feature in any system it is not generally a necessity, and system performance is neither improved nor reduced, although natural language systems might be able to provide better explanation at a level required by the user, rather than the simple, pre-programmed text currently offered by most systems. This is likely to be an important feature with future generation expert systems since investigations into the nature of explanation in human

expert interactions has revealed that the 'user' requires various different types of explanation (Kidd 1985).

Coupled to the MMI of the system in some cases is a tutorial type module. This module is normally used to pass on both the knowledge required to perform in a domain, and the strategy with which that knowledge should be applied. The idea to use rulebased systems for tutoring of students has been demonstrated in the GUIDON system (Clancey 1979) and also in the NEOMYCIN system (Clancey 1981, 1983), both of which were designed to use the MYCIN rulebase to teach medical diagnosis.

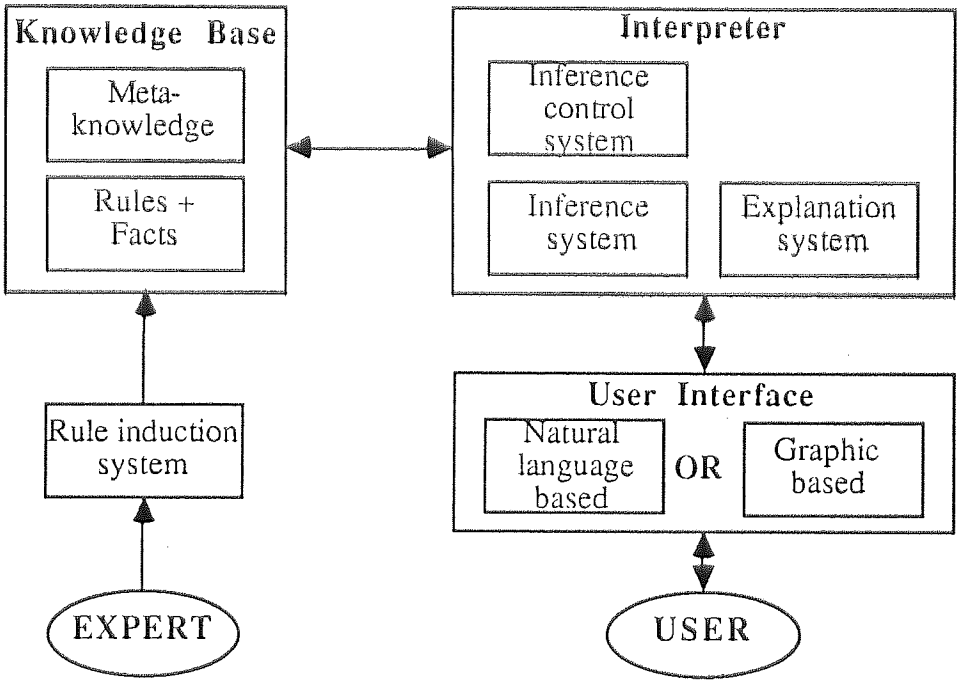

Figure 2.3 Diagram of the 'ideal' expert system with intelligent interfaces and learning capability.

2.2.2 Representation of Knowledge for IKBS use

AI research into knowledge representation schemes has produced a wide variety of techniques for modelling human intelligence and knowledge utilization. Much of this research has been conducted by psychologists interested in cognitive studies of human reasoning and memory techniques. This research has proven particularly valuable in the

design of systems for natural language processing and understanding. Within the IKBS community only a restricted set of the representation schemes, now under consideration in the wider discipline of AI, are to be found in regular use. This restriction of knowledge representation schemes does not reflect a lack of research within the field of IKBS but rather reflects the fact that the representations already in use fit the major requirements for producing a working system, while also producing code which the expert can understand, and behaviour which at least appears intelligent to the user.

2.2.2.1 Knowledge Representation With Production Rules

The most widely used representation scheme for IKBSs is that of production rules described by Davis *et al.* (1977). It is this representation which was used in the major, early 'expert system' developments for example MYCIN (Shortliffe 1976) and R1 (McDermot 1982a). With such a solid foundation in what can only be described as the classic research systems, it is hardly surprising that production rules have found such popularity with expert system programmers and have become the standard representation for commercial shells.

Production rules are basically inference rules which can be presented in a simple notation as:

IF < premises >
THEN < conclusion >

On evaluation of such a production rule if the premises evaluate to be true, that is the logical combination of conditional statements evaluates to be true, then the conclusion is assigned as being true and is added to the list of known facts. However, if the premises evaluate to false then either the conclusion is assigned as false or the rule is completely ignored depending upon the specific behaviour of the interpreter in use. In general the truth values of propositions in expert systems are not represented by boolean values but

rather by some numerical measure of belief. Such measures of belief are often related to probability theory and allow the system to reason with incomplete knowledge or data. The reasons behind numerical measures of belief, and methods for handling such measures of belief are detailed later (Section 2.2.4).

Reasoning Strategies - with Reference to Production Rules

Reasoning system architectures can normally be classified into one of two basic categories based upon the goal solving strategy employed by the interpreter. In most systems this strategy is either 'forward chaining' or 'backward chaining', although some more sophisticated hybrids do exist (Quinlan 1983, Mellish 1985). A brief outline of these important control strategies is given here. Any reader requiring a more detailed discussion is directed to any of the good introductory AI texts (e.g. Bundy 1980, Hunt 1975, Rich 1983, Winston 1984).

Forward Chained Reasoning Systems

Forward chained reasoning is a data driven process, that is, the process of deriving conclusions is carried out by using all of the data available to the system. For clarity in demonstrating the operation of a simple forward chained production system, it is better to restrict the measures of belief to a boolean system. A simple interpreter loop can then be imagined which scans the rules in sequence checking if any fires, that is their premise evaluates to true. If a rule fires then its conclusion is made true, the rule is deleted or prevented from firing again, and the loop is repeated otherwise the next rule is tested. Should no rule be able to fire then unless further data is available the programme stops. This interpreter could then be provided with a set of production rules and data as shown in Figure 2.4.

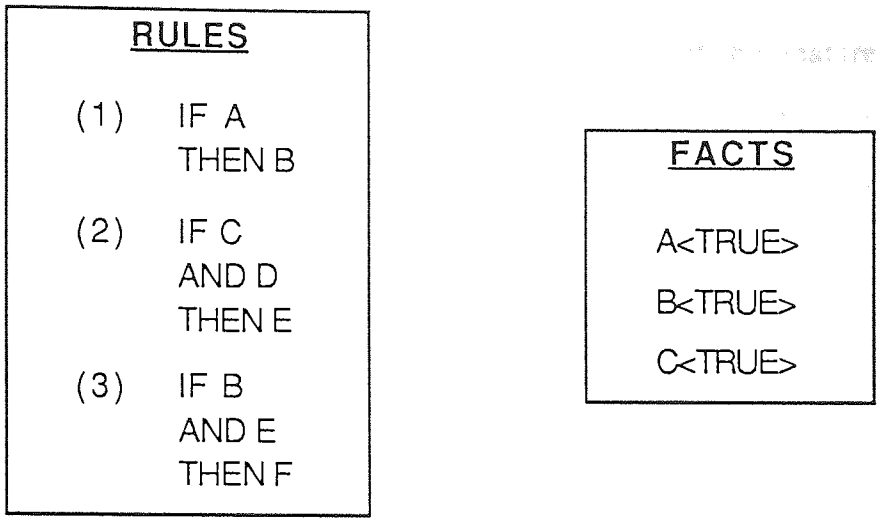

Figure 2.4 Simple three rule knowledge base and associated facts.

To use this knowledge base the interpreter will scan the rules and find that Rule 1 fires so that B<TRUE> is added to the available data. Now Rule 2 fires adding its conclusion E<TRUE> to the data set. Finally Rule 3 is able to fire using the new data and F<TRUE> is added to the data. In this way production rules are used to carry forward a chain of inference allowing new data to be created based on raw data input to the system. In a slightly more useful system the interpreter could be given some goal to find so that when the correct data is added the system could stop and report its findings.

The problem with this type of strategy is that the interpreter is allowed to make all possible inferences from some piece of data in a generally unconstrained manner. Although such a strategy is very desirable, in the sense that the consequences of new data are immediately propagated through the inference chains, a less desirable facet of the strategy is that data is derived which may have no bearing on the goal to be solved. This can be demonstrated if the rulebase in the example system were defined as shown in Figure 2.5.

It is clear that the system would infer X<TRUE>, W<TRUE> and Z<TRUE> even if it were only required to find out about the proposition F. This type of system behaviour falls within the umbrella of the term 'combinatorial explosion', which is used to describe the problem where many propagation paths exist for the system to follow, but only a small

subset of these are relevant to the goal to be solved. Even if this feature of forward chaining systems can be tolerated in an application, it is still likely that the system will require to gather all possible data in order to satisfy its goal, much of this data being likely to prove completely irrelevant. However forward chained reasoning has been used in producing expert systems with some success, particularly in situations where only a small amount of input data is used.

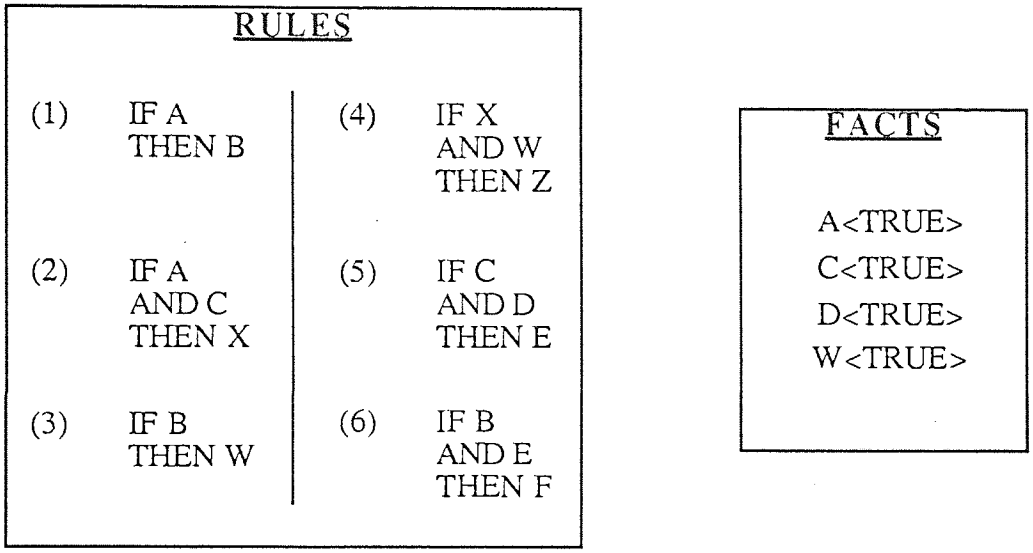

Figure 2.5 Simple six rule knowledge base and associated facts.

Backward Chained Reasoning Systems

Backward chained reasoning strategies solve the problems of irrelevant data collection and inference found in forward chained systems. Unfortunately some new problems become apparent as the cost of this new strategy. Basically, a backward chained system, instead of scanning rule premises to find those which fire, scans the rule conclusions to find those relevant to solving the goal in question, steadily reducing the problem to a set of simpler goals. This can be demonstrated using the same rule base example as for forward chaining. In this case however, the system is requested to find out whether F<TRUE> can be ascertained. Instead of using the rule premises, the interpreter

examines the conclusions to find one mentioning F. Rule 6 is found to infer F but requires B and E to be known, so now the interpreter attempts to solve one of the sub-goals B or E. Let us assume that E is chosen, the system will then find that Rule 5 can be used to solve E if the goals C and D are solved. Since no rules are available the system will look to the user for the solution to these goals. In this way a backward chained system is capable of only following inference chains which are likely to prove useful in finding a solution to the desired goal.

Unfortunately problems do exist with this strategy since, given more than one goal to choose from, for instance which of F or Z is true, it is possible for the system to work back through one inference chain and to find that this chain is unproductive. Many questions will have been asked of the user, often in an unorganized manner, before the system changes its tack to ask questions in a new context, thus appearing completely undirected and unintelligent to the user. This problem is particularly true at the beginning of a consultation where the system has little or no evidence to direct it away from solutions which are unlikely. In spite of this limitation backward chained reasoning is in most cases the accepted strategy for use in inference engines, for example this is the strategy employed by MYCIN. More recent systems however, employ a more complex set of controls to allow the system to maintain some consistency in its questioning of the user. These controls are discussed in detail later (Section 4.1.1).

Hybrid Control Strategies

Hybrids of the two solution search strategies are used in some systems and allow some improvement in performance. For example PROSPECTOR is forward chained when initial data is offered, but continually switches back and forth between this and a backward chaining algorithm, similar to that of MYCIN, in order to direct further questioning. By using this combined strategy the system is prevented from following a chain of reasoning once it has proven fruitless and is able to select the most interesting solution path based on

the highest degrees of belief. A good description of the advantages to be gained from a hybrid system is given by Mellish (1985) who describes a system which he has implemented as a teaching prototype in the POPLOG programming environment. This teaching system is detailed later (Section 4.1) and is also discussed in one of the POP11 text books (Ramsey & Barrett 1987).

Granularity of Knowledge Represented as Production Rules

One of the major reasons for the success of production rules as a knowledge representation scheme is the embodiment of recognizable pieces of domain knowledge within a rule. This feature means that the expert responsible for writing and maintaining the knowledge base always knows the meaning of any rule as written and also normally finds the task of formulating new rules much more straight forward. Using this idea of recognizable knowledge units causes a 'granularity ' to the knowledge base, with the level of detail used in a rule being termed the grainsize of the rule. In this way the more low level a rule conclusion becomes, the less is the grainsize of the rule. For example in geology we might have a set of rules as in Figure 2.6.

IF the rock shows a slaty fabric
AND the rock is very fine grained
THEN the rock is a slate

IF the rock is fissile
AND the fissility is pervasive
THEN the rock shows a slaty fabric

IF the rock is able to be split into sheets
THEN the rock is fissile

IF platy minerals show strong preferred orientation
THEN the rock is able to be split into sheets

Figure 2.6 Series of geological rules showing varied levels of abstraction.

These rules can be seen to represent granular units of knowledge but the grainsize of the rules decreases as the rules become steadily more detailed and much lower in level. In fact if we allowed a numerical certainty value on these rules they become much less certain as we move toward inferring information which the user ought to be able to give us.

2.2.2.2 Semantic Nets as a Knowledge Representation Scheme for Inference

Semantic nets are extensively used in the field of natural language understanding as they form a very natural method of linking concepts together. Basically a semantic net is a group of meaningful concepts (nodes) which are individually linked to other related concepts, the links (arcs) representing the nature of the relationship between nodes or the procedure by which nodes are related. A typical example of a very simple semantic net would be the representation of the hierarchical relationships of minerals as shown in Figure 2.7.

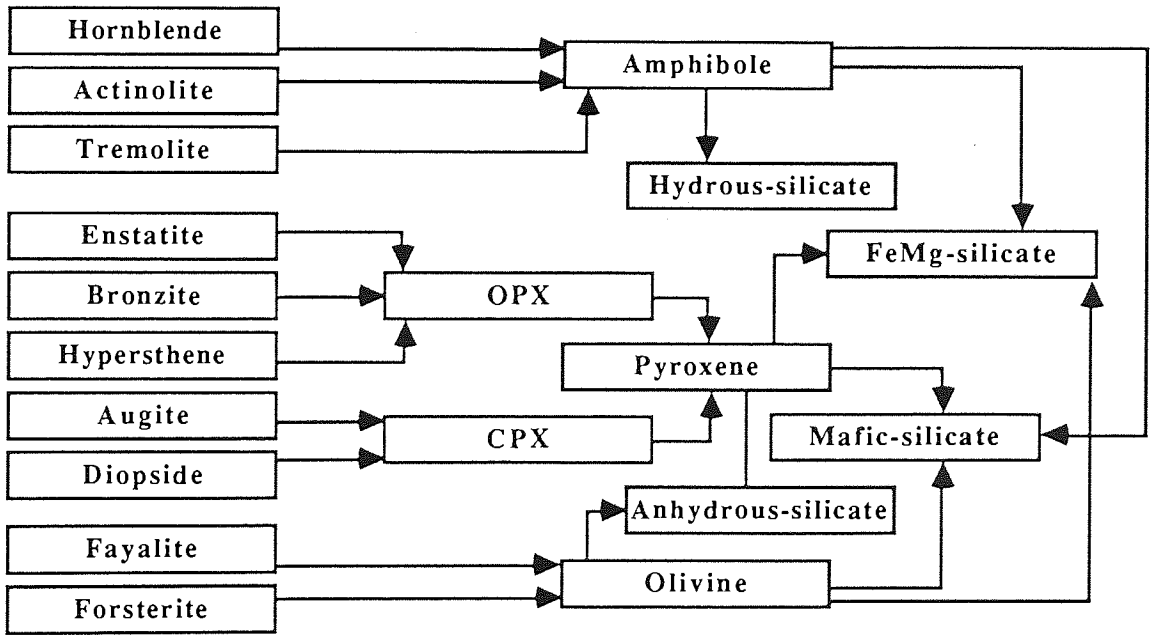

Figure 2.7 Sample portion of a semantic type net showing the relationships between mineral terms as simple is_a type links.

To make use of a semantic net representation for inference it is necessary to allow complex sets of information to be attached to both the nodes and their links. One early system which used a semantic net type representation is PROSPECTOR (Duda *et al.* 1978) which represents the English form of its knowledge as production rule like structures which are used to build an internal representation as a semantic net, or dependency graph.

A more complex net representation was used in IRIS (Trigoboff & Kulikowski 1977, Trigoboff 1978). IRIS was designed in order to experiment with new knowledge representations for use in clinical diagnosis. In IRIS the nodes of a semantic net are used to represent the clinical concepts of use in diagnosing disease, for example the diseases possible, the symptoms shown and the diagnosis chosen. These concept nodes are then linked together to show their relationships to one another, examples of the types of link being 'causes','treatment for', and 'associated with'.

In order to make use of the semantic net IRIS has a set of Information Specifications (ISPECs), attached to each node, which contain the data to be associated with the node during a specific consultation, and are able to be created, deleted or modified during a consultation. ISPECs are stored as records in the format shown in Figure 2.8.

NODE the concept node to which an ISPEC belongs

SIDE the side of the body to which the ISPEC refers, for example right or left kidney. Concepts such as migraine have a NIL value.

MB measure of belief in the concept node

TIME the time period over which theISPEC is valid

MODS variables with values

TYPE tells the interpreter how the ISPEC is to be used

Figure 2.8 Layout of a typical ISPEC as used in the IRIS inference system indicating the entries required for each node.

Thus the statement 'the patient had a previous severe migraine headache from October 5 87 to October 6 87' could be represented in an ISPEC shown in Figure 2.9.

NODE = Migraine-headache

SIDE = nil

MB = [1 0]

TIME = [Oct 5 87 - Oct 6 87]

MODS = Degree : severe

TYPE = Patient-history

Figure 2.9 Example ISPEC for a specific instance of a 'migraine headache' as it might be represented by IRIS.

Once the ISPECs are created for a node it is necessary to be able to propagate the information to other related nodes. This propagation is carried out by the interpreter using decision tables associated with the links between nodes. The decision table is used to decide whether the ISPECs on the propagating node contain the information to allow creation or modification of ISPECs on the target node. A very important feature in IRIS is the "cone" where multiple nodes, at the base, are required to infer a new node, at the apex. Cones are built by attaching a single decision table to all of the links required for the inference. Decision tables such as this are the 'production rules' in IRIS's knowledge base and allow IRIS to perform actions such as creating or modifying ISPECs and updating degrees of belief. When propagation begins in this way it proceeds as a wave through the inference net until the effects of a piece of data have been fully evaluated in a forward chained manner.

2.2.2.3 Frame and Model Based Representation Schemes

Frames are one of the popular data structures of AI and appear to have a considerable amount of psychological validity from research into script based memory (Schank & Abelson 1977). A frame is basically a conceptual unit of knowledge with slots to identify the data associated with the concept as well as the relationships between frames. Slots can then have default values associated with them in order to represent expectations in the data so facilitating the detection of conflicts and the recognition of special cases. An example is shown in Figure 2.10. A valuable feature of frame representations is in encoding factual data where the use of inheritance in hierarchical data structures makes alteration of data simple. For example the concept of a person could be encoded as a 'person frame' including all of the facts normally associated with people: age, date of birth, address, physical characteristics, etc. If we create many instances of the person frame and then decide to change some default feature, say number of arms, then instead of having to alter every instance, we only require to change the main person frame. Normally in frame systems inheritance can be from many different parent types so that a particular company manager might inherit from a person frame, a managers frame and a golfer frame. If we subsequently alter the default characteristics of any of these groups then the data on our particular manager will be automatically updated. This feature of inheritance makes frame representations very flexible and simple to maintain.

One of the main advantages of representing knowledge in models or frames is that the system will be focused onto a certain area of the knowledge base while a frame is checked so that the questioning of the user is made to appear intelligent. Well described frame based systems are WHEEZE (Smith & Clayton 1980, 1984), PIP (Szolovits & Pauker 1978) and INTERNIST (Pople 1977).

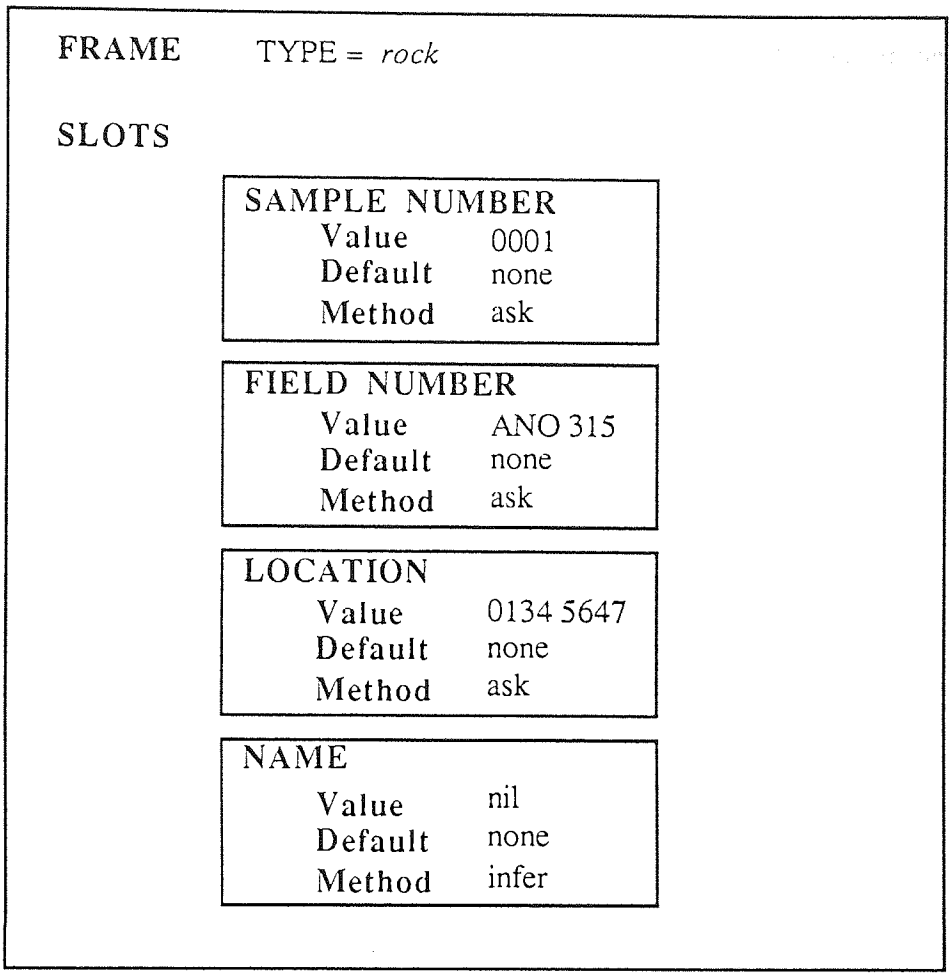

Figure 2.10 An example of how a frame representation might represent the typical rock specimen.

WHEEZE is a re-implementation of the PUFF rule based system, for the diagnosis of Pulmonary Function Disorders, designed as a test bed for frame-based representation. WHEEZE represents all patient data and expert knowledge in a selection of three frame types. The most important type of frames in WHEEZE are assertion frames. These contain information relating to some hypothesis or proposition of importance to the system. Assertion frames contain information as shown in Figure 2.11.

The other important frame types in WHEEZE are Patient Frames and Patient Datum Frames. Patient frames are used to store all of the data relating to a particular patient and consultation and form the basic database on which WHEEZE operates. Patient datum frames are used by WHEEZE to store values for particular variables applicable to an individual patient. WHEEZE uses these frames to store methods for finding values, for

example the text to be used for questioning the user, along with information on expected values and assertions which are categories of the datum.

ISA	assertion
DESCRIPTION	descriptive information about the assertion
MANIFESTATION	conditions upon which this assertion depends, a variety of functions make linking manifestations for inference possible
MANIFESTATION_OF	a list of assertions which are influenced by this one
CERTAINTY	certainty of this assertion if manifestation is true
SUGGESTIVE_OF	other related assertions to be considered if this one is true
COMPLIMENTARY_TO	other assertions to consider if this one is false
CATEGORIZATION_OF	patient datum which is categorized by this assertion
CATEGORIZATION_CRITERION	possible values for the datum
D_O_B	measure of belief for a particular patient
FINDINGS	report text for use when an assertion is believed by the system

Figure 2.11 Example of the typical frame layout as used in knowledge representation in the WHEEZE system.

The control strategy of WHEEZE is agenda based with the system investigating the top assertion on the agenda. If the Manifestation slot can be evaluated then if the assertion under consideration is shown to be true, the assertions listed in the 'SuggestiveOf' slot are added to the agenda, otherwise the assertions in 'ComplimentaryTo' are added. If however the 'Manifestation' cannot be immediately evaluated then the unknown assertions are added to the agenda and dealt with according to their importance.

It is possible to tune the control of WHEEZE by altering the values of importance assigned to assertions which are manifestations and to assertions which are listed in

'SuggestiveOf'. In this way the knowledge engineer can allow the system to have the ability to alter its strategy depending upon data coming in. This is important for instance when a piece of unexpected data arrives causing some new hypothesis to be immediately investigated to account for the data.

Although each of the representations described above is generally treated as a distinct method in the literature, it is clear that they all share certain features. In each case some form of inference mechanism is required and in every case this is a variation on the production rule. It is also clear on close examination that the representation of IRIS and that of WHEEZE are very similar in nature although the operation of each is different. For example a node in IRIS has a set of ISPECs attached to it along with a set of links to other nodes, each link holding the rules for propagation. This is almost the same as the case of WHEEZE which has the assertions (nodes) stored as frames containing slots for linked assertions and for 'rule' type expressions to be used in propagation.

2.2.3 High Level Control Strategies in Inference

One of the earliest problems found with pure production rule systems centred on the strategy employed in selecting rules to use. In the example production rule interpreters described above (Section 2.2.2.1) the system simply checked rules in the order in which they appeared. In reality this is not a very intelligent strategy since the system will appear to be asking questions randomly of the user. It is possible to remove this random questioning by careful ordering and tuning of rules, but this tends to fall apart on specific cases which do not closely correspond to those used in the tuning.

2.2.3.1 Meta Knowledge

Davis & Buchanan (1977) attempted to solve the control problem with the use of meta-knowledge whilst also providing the system with the ability to understand its own knowledge. In their paper Davis & Buchanan describes two forms of meta-knowledge,

domain independent knowledge about knowledge itself and domain specific knowledge about how to apply knowledge within the domain. Both of these forms of meta-knowledge are of interest in IKBS design.

Domain independent meta-knowledge has been applied by Davis in the TEIRESIAS system in order to allow interactive transfer of knowledge to the system. TEIRESIAS uses this knowledge to examine the contents of rules already in the system in order to compare this with knowledge being added. This allows TEIRESIAS, described in detail by Davis and Lenat (1982) and Davis (1984), to construct expectations as to what clauses might also be added to the new rule and how that rule is likely to interact with other rules. TEIRESIAS operates by way of the expert testing the rule base and if he disagrees with the result he then attempts to discover the fault with the system's help. To do this the expert asks why a particular solution was not chosen prompting the system to discover rules which might have applied and why they did not fire. If this is because a rule is missing then the expert is asked to enter the rule and then edit it if necessary. TEIRESIAS then checks this rule with others in the rule base and if it finds that normally two clauses appear together and one is missing in the new rule it will ask the expert directly about this. Assistance of this type becomes very important in large knowledge bases. For example, R1 (XCON) is reported to now contain some 5000 rules and according to McDermot (1986) has become extremely difficult to maintain with regular unrecoverable failures in the knowledge.

A further use of such meta-knowledge is in allowing the system to understand the contents of its rule base. This in turn allows the system to degrade more gracefully when a consultation example does not fit any of diagnoses or classifications available to it. In other words the system knows the limits of its knowledge as a human expert does. This is particularly advantageous as the performance of standard expert systems tend to degrade very sharply when the limits of their knowledge are reached.

Domain dependant knowledge is of slightly more interest in respect of its use in controlling the consultation strategy of the system. It is in fact this form of meta-knowledge which is most often referred to in the literature. Davis suggested that in order to improve the behaviour of a production rule system it is necessary to have rules which are able to selectively invoke different areas of the knowledge base. These rules behave in the same way as the lower level inference rules in that they only fire when the correct data is available. In this way low level rules are only invoked if they are likely to be of some use in a particular consultation. Once a system is capable of handling such rules the expert and knowledge engineer are able to construct meta-rules, as they are called, to control the complete rule base at different levels, that is meta-rules controlling the invocation of other meta-rules.

One important problem in such a system of rules occurs where it becomes increasingly difficult to know what is knowledge and what is meta-knowledge in the sense of control knowledge. In order to simplify the distinction, between low level inference knowledge and meta-level control knowledge, researchers have experimented with complex representations. As described above (Section 2.2.2) frame and augmented semantic net systems are in many ways similar in that all associated assertions or propositions are related explicitly. In this way such systems provide a mechanism for the control of inference, which is strengthened in the case of WHEEZE by the addition of importance factors to the various 'links'. Since each scheme also uses some form of 'rule' to allow propagation then the frame or semantic net node provides a form of meta-knowledge.

One of the influences behind the WHEEZE system, also developed as an experiment after PUFF, was designed to test the value of a combined frame and production rule representation. This system, known as CENTAUR (Aikins 1980, 1984), was developed in an attempt to make use of the strengths of each of the two independent representation schemes. In CENTAUR the frames, or prototypes as Aikins calls them, are used to encode the control information. In this way the benefits of a frame system, from the point

of view of focusing, are gained along with explicit control data. By also using production rules however, it is possible to represent the lower level inference knowledge in a more natural manner and more clearly than with say either of the frame or semantic net representations. In CENTAUR the frames indicate what data has to be gathered to prove a prototype along with how that data should be gathered. Once a prototype has been confirmed the system applies IFCONFIRMED or IFDISCONFIRMED slots as a guide to which new prototypes should be considered, these slots being the equivalent of Davis' meta-rules as shown in Figure 2.12.

IF this frame is proven
THEN consider <frame 1>, <frame 2>

IF this frame is disproven
THEN consider <frame 3>, <frame 4>

Figure 2.12 Equivalent meta-rule translation of control information represented in a frame based control system.

Thus the control of the system is moved to another prototype. During the execution of a prototype CENTAUR may gather a piece of data which causes a trigger rule to fire, immediately moving control to a new prototype which might be more useful.

2.2.3.2 Blackboard Systems

Research in IKBS technology has led to the acceptance that no single knowledge representation can at present handle the knowledge from every domain of human expertise. This is becoming clear even in the commercial shell developments many of which now provide multiple representations and multiple knowledge sources. In order to control inference in truly integrated multiple representation systems researchers have used the 'Blackboard Architecture' developed from the Hearsay II speech recognition system (Erman *et al.* 1980) and described in detail by Nii (1986a, 1986b) in a series of two papers on architecture and use. Such architectures allow systems to integrate knowledge from a variety of sources in order to solve a particular problem. An example scenario is shown in

Figure 2.13. The architecture essentially consists of a set of knowledge sources which can each solve some particular aspect of a problem, a scheduling system and a 'blackboard' on which problems and solutions are posted. When a problem arises some triggering mechanism chooses the knowledge sources which are able to solve that problem. The scheduler then allows the knowledge source to assume control and attempt to reach a solution to the particular problem. This may involve returning to the scheduler with some new problem to be solved which triggers a further knowledge source to contribute to the solution (see Figure 2.13). Hayes-Roth (1985) describes the variety of complexity in the control strategies of blackboard systems. She states that the scheduler may, in sophisticated systems, be able to dynamically alter the strategy based on the reliability or cost of competing knowledge sources in solving problems, whilst in less sophisticated systems the scheduler is restricted to a pre-defined strategy.

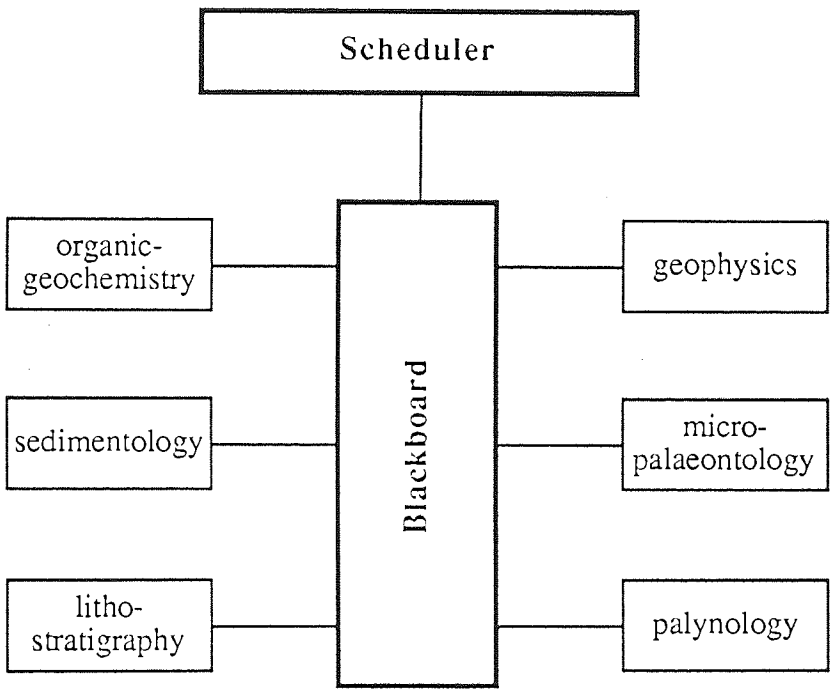

Figure 2.13 Example showing how a blackboard system would simulate the human technique of using a conference of experts from different fields to solve a problem.

2.2.4 Measures of Belief in IKBS

Throughout this chapter reference has been made to measures of belief, with respect to the different systems reviewed, without detailing the reasons for and methods of calculating the measure of belief. Many researchers have found that in expert domain knowledge, which relies largely upon experience, the rules which are used for inference are not completely certain. That is the rules are only 'rules of thumb' or more correctly heuristic rules. This uncertainty associated with the rules in a knowledge base must be modelled some how if the system produced from the rules is to appear intelligent or produce the same results as the expert. Coupled to this uncertainty in rules, systems must also be able to handle uncertainty in input data. This form of uncertainty derives from three sources:

1. Uncertainty on the user's part as to whether some fact is true or not.

2. Incomplete data, where users simply are not able to gather a required piece of data, either because they do not know how, or the data is not available.

or

3. Uncertainty in the meaning of a term, for example the term monomineralic may mean that one mineral accounts for more than 80% of the rock. Yet if it was 79% we would not say that the rock was not monomineralic with complete certainty.

Many mathematical methods for handling such uncertainties have been produced by IKBS researchers, each method inspiring variable amounts of support or criticism from the IKBS and AI communities. The most important methods which have been suggested for use in expert systems are Bayesian probability (Duda *et al.* 1978), ad hoc certainty (Shortliffe & Buchanan 1975), Dempster-Shafer theory (Shafer 1976) and Fuzzy Set Theory (Zadeh 1978, 1985). Of these only the Bayesian and ad hoc systems have been applied on a widely successful basis, although researchers are beginning to investigate the

application of Dempster-Shafer (Gordon & Shortliffe 1984, 1985) and Fuzzy Set Theory (Ben-Bassat 1985, Anderson *et al.* 1985, Applebaum & Ruspini 1985).

2.2.4.1 Bayesian Probability Theory

Probability theory is the basis of one of the most important methods used in handling uncertainty in expert systems. One of the most important systems which was built around a probabilistic reasoning system is PROSPECTOR, which is based on Bayes' theorem (Duda *et al.* 1978) and is well described in Alty & Coombs (1984). The advocates of Bayes argue strongly that this technique for dealing with uncertainty is based on a sound mathematical theory and can therefore be clearly justified. However, there are several problems which occur in using Bayes which must be dealt with. Many of these problems can be highlighted with reference to the methods used in the PROSPECTOR implementation of Bayes, but firstly a brief introduction to the theorem is required.

Bayes' theorem allows the user to relate probabilistically some conclusion with some piece of evidence so that in a production rule say, the probability of the conclusion can be related to the probability of the premise clause. Bayes' theorem is given in Equation 2.1 where $p(H_i|E)$ is the probability of Hypothesis H_i based on the evidence E being true

$p(E|H_i)$ is the probability of the evidence E based on the hypothesis H_i being true

$p(H_i)$ is the 'a priori' probability that Hypothesis H_i is true

$$p(H_i|E) = \frac{p(E|H_i) * p(H_i)}{\sum_{j=1,n} p(E|H_j) * p(H_j)} \qquad 2.1$$

Where $\sum_{j=1,n} p(H_j) = 1$

In order to allow simple implementation of Bayes' theorem the probability function of Bayes rule (Equation 2.1) is rewritten in terms of odds where the odds of a hypothesis $\emptyset(x)$ is given by Equation 2.2.

$$\emptyset(X) = \frac{p(X)}{1.0 - p(X)} \qquad 2.2$$

So that Bayes' rule can be rewritten in the form:

$$\emptyset(H_i|E) = LS * \emptyset(H_i)$$

$$\emptyset(H_i|\sim E) = LN * \emptyset(H_i)$$

where $[LS = p(E|H_i) / p(E|\sim H_i)]$ and $[LN = p(\sim E|H_i) / p(\sim E|\sim H_i)]$

From this it can be shown that the relationship of LS and LN should follow a set of simple rules.

If $LS > 1$ then $LN < 1$

If $LS < 1$ then $LN > 1$

If $LS = 1$ then $LN = 1$.

It is found in practice that most inference rules propagate positive beliefs so that the value of LS is normally $>> 1$ and the value of LN should be < 1. Unfortunately the expert prefers to write rules where the value of LS is $>>1$ but the value of LN is 1, that is rules where the evaluation of the premises to true gives a change in the probability of the conclusion while an evaluation of the premises giving false produces no change in the probability of the conclusion. In order to allow this, systems which are based on Bayes' theorem have followed the lead of PROSPECTOR in making interpolations for the probability of a conclusion using LS and LN when the change of probability in the evidence is positive and negative respectively. In this sense Bayes' theorem is not strictly adhered to.

Some problems exist in attempting to use Bayesian methods since firstly the rules are bent slightly to allow the expert to represent rules in the way in which he intuitively believes them to work, and secondly Bayes' theorem requires a set of a priori probabilities to be attached to all of the hypotheses in a system. This is often a very difficult requirement for the expert to satisfy since in many domains no such quantitative data is

available and certainly in geology any a priori probabilities would be geographically dependent. A third difficulty with Bayes is that to be strictly correct the hypotheses in the system must be completely independent, a rule which is in fact impossible to satisfy in real world systems.

2.2.4.2 MYCIN type Certainty Factors

In MYCIN, degrees of belief are handled using a number system designed to model what the expert means when defining a rule (Shortliffe & Buchanan 1975, Shortliffe 1976, 1984). The system chosen is described as 'ad hoc', since it does not directly correspond to any of the mathematical theories of probability, although Gordon & Shortliffe (1980, 1984) have shown MYCIN certainty to be a special case of the Dempster-Shafer theory of evidence (Section 2.2.4.3) and other workers have attempted to rationalise the model in terms of probability (Adams 1984, Heckerman 1986). In the MYCIN system Shortliffe uses a number on the scale -1 to 1 to represent the certainty of a piece of information. On this scale -1 represents complete evidence against a hypothesis while 1 represents complete certainty for the hypothesis. For example B<TRUE> and B<FALSE> would be represented as B(1) and B(-1) respectively. Zero represents the special case of having no evidence for or against a hypothesis, while numbers between 0 and 1 represent varying degrees of belief for a hypothesis, and numbers between 0 and -1 represent degrees of belief against the hypothesis.

Certainty Factors (CF) are in fact composite numbers formed by the combination of the evidence for and that against a hypothesis, Measure of Belief (MB) and Measure of Disbelief (MD). In early versions of MYCIN the three numbers are related by the equation:

$$CF = MB - MD.$$

In MYCIN the data of the system, that is the hypotheses used by the system, are stored as associated triples of the form:

<attribute> <value> <tally>

where attribute is some variable used by the system and value is one of the values attached to that variable. Tally is the CF with which the system believes that the value given to the attribute is the correct value. For example a MYCIN type system might represent its beliefs about the data associated with minerals in a rock, based on current evidence, as shown in Figure 2.14.

Attribute	Value	Tally
colour mineral1	green	1.0
colour mineral2	colourless	1.0
ident mineral1	augite	0.8
ident mineral1	diopside	0.3
ident mineral2	plagioclase	-1.0
ident mineral2	quartz	0.85

Figure 2.14 Example attribute value table as used in a MYCIN type system to represent working data.

Certainty measures are propagated across MYCIN inference chains by a series of combination functions and inference rules. Within a production rule the expert may wish to represent uncertainty in the rule itself. This is done by attaching a modifier to the rule conclusion such that the certainty of the evidence used in the rule is multiplied by this modification factor to evaluate the certainty of the conclusion. To find the certainty of the evidence within a production rule MYCIN uses a simple set of rules of inference as follows:

1. Conjunctive clauses should be evaluated according to the minimum certainty of their individual clauses.

2. Disjunctive clauses should be evaluated according to the maximum certainty of their individual clauses.

Once the complete premise clause has been completely evaluated the resultant figure is multiplied by the certainty of the rule to attach a measure of certainty to the conclusion, this figure then becoming available for propagation through the premises of the next rule in the inference chain. This part of the MYCIN certainty handling system causes very little problem for other workers, but the system does come in for a lot of criticism on the methods used to combine evidence from multiple rules making the same conclusion. To allow such combination of evidence, MYCIN uses a group of three functions depending upon the nature of the pieces of evidence being combined.

IF A	IF D
AND B	AND E
THEN(1.0) C	THEN(-1.0) C

Figure 2.15 Example of rules combining their evidence allowing MYCIN to reach a degree of belief in an hypothesis C

Consider where two rules are used to evaluate a hypothesis C as in Figure 2.15. If (A AND B) evaluates to give C(0.4) while (D AND E) evaluates to give C(0.3) then MYCIN will combine these values using the function in Equation 2.3.

$$CFcom = CF1 + CF2 - (CF1*CF2) \qquad 2.3$$

This produces a value for CFcom of 0.58 for the hypothesis C. If the rules had evaluated as Rule 1 C(-0.4) and Rule 2 C(-0.3) then MYCIN would similarly combine these values, but this time with the function in Equation 2.4.

$$CFcom = CF1 + CF2 + (CF1*CF2) \qquad 2.4$$

This time a resultant certainty of -0.58 is produced for the hypothesis C. These equations are directly equivalent to the probabilistic sum equation and can be defended on these grounds. The area in which the calculation methods are questioned is when two or more rules are combined to produce one normalized CF. This combination was carried out in early versions of MYCIN by use of the function:

$$CFcom = MB - MD$$

In later versions however, this function was seen to produce undesirable results. Several pieces of evidence might, for example, combine to give a CF around 0.98. If *one* piece of denying evidence now occurs with a CF of -0.6, then the combined certainty will reduce to 0.38. In light of this behaviour the combination function for combining positive and negative evidence was altered by redefining CF (Buchanan & Shortliffe 1984 p216) to the form shown in Equation 2.5.

$$CFcom = (MB - MD)/(1 - \min(MB,MD)) \qquad 2.5$$

With this new function, the effect of combining the CF(.98) value, equivalent to MB(0.98), with the CF(-0.6), equal to MD(0.6), is a value of CF(0.95). This function for combining evidence from multiple rules produces results which are more intuitively correct but mathematically unfounded.

One other area of the MYCIN system which is of interest is the fact that all rule conclusions are assumed independent. This worries many workers in the field. The feature was added to MYCIN to allow the system to come up with a set of correct diagnoses, as it is possible in MYCIN's domain to have more than one correct result. Many domains are therefore not suitable for application of the MYCIN certainty scheme since the hypotheses in the domains are members of exclusive sets and are therefore not independent. MYCIN type certainty handling is useful for domains with independence of hypothesis and also does not require complete prior probabilities for the hypothesis as in the Bayes' theorem used in PROSPECTOR. This can be a major advantage towards producing expert systems where the expert is unable or unwilling to provide statistical data for the provision of more accurate probabilities.

The domain of metamorphic classification, for example, is one where more than one rock name may be correct with a given set of evidence and also a strict probability distribution for the possible rock classes would be of little use. The probability problem

might be highlighted here by supposing that the system is to be coded with the probabilities of rock types on a global scale. If the system is then used in a field area where only two, normally rare rocks, are likely, then the results from the prior probabilities are incorrect. This could be overcome by allowing the system to learn that the probabilities are different and to self adjust them. An approach similar to this was taken in Simons' chequers program but this is reported to adjust its heuristics erroneously when playing a poor human adversary (Winston 1984). Such a behaviour might then be expected of our geological system since the system would learn and adjust its probabilities from the restricted field area, and carry these over to any new field areas.

2.2.4.3 Dempster-Shafer Theory of Evidence

As with the MYCIN system, the Dempster-Shafer (DS) theory of evidence does not require any accurate prior probabilities calculated for its operation. What this system does involve however is a more accepted mathematical theory for the combination of evidence. DS also tackles the assumption of independence of hypotheses made in MYCIN and allows for mutually exclusive hypotheses having direct effects upon each other. For example if some entity 'ENT' can only be one of three colours say red, green or blue, then if we know that 'ENT' is green with a belief of 0.6 then we also know that the maximum likelihood of 'ENT' being red or blue is 0.4. DS allows for the propagation of probabilities across sets in this manner and can adjust the beliefs of members in subsequent subsets accordingly.

In DS theory a hypothesis has two values attached to it, giving a range for the probability of the hypothesis. This range can be used as a measure of the uncertainty in the hypothesis, that is the extent to which it has been evaluated, as well as a measure of the probability of the hypothesis, that is the degree of belief in the hypothesis. The DS numbers are represented as a pair on the range [0,1], for example [s(H),p(H)], where s(H) represents the evidential support for H, that is the degree to which H is shown by the

evidence, and p(H) represents the degree of plausibility of H, that is the degree to which H may potentially be shown. The value p(H) can also be taken to represent the degree to which H is NOT doubted since the value 1-p(H) is equal to s(~H) or the evidential support for the hypothesis (not H). In many systems it is in fact the values s(H) and s(~H) which are stored for use. In this DS notation it should be noted that it is possible to show complete ignorance about a hypothesis with a range of [0,1], whereas in a purely Bayesian approach some distinct prior probability would be attached to the hypothesis. For example if two hypotheses A and B are taken as mutually exclusive and exhaustive then in a Bayesian system they might carry individual probabilities of 0.5. In DS however, each of A and B would initially hold evidence ranges of [0,1].

The DS theory for combining evidence is derived from set theory, so that the relationships between hierarchically related hypotheses are easily maintained. All individual hypotheses are subsets of a given set, often referred to as a frame of *discernment*(@). Using Dempster's rule, evidence about any hypothesis is used to evaluate all related hypothesis in @ since all members of @ are evidentialy related. This provides an 'evidence distribution' or more properly a 'probability mass distribution' for the members of the set @. If more than one 'mass function' is to be calculated for a set then the individual functions are combined by the use of an 'orthogonal sum' to produce a unified probability mass distribution. Dempster's rule as it stands requires a great deal of computational overhead if it is to be implemented as a method for handling uncertainty in Expert Systems and as yet no system has used a complete implementation to propagate uncertainty (Batnagar & Kanal 1986), although some systems do use DS as part of their Uncertainty model (Garvey *et al.* 1981).

However, in their 1985 paper, Gordon & Shortliffe suggest a restricted version of DS which can be operated successfully if the frames of discernment are coded in a strictly hierarchical manner. This cuts down drastically the amount of computation required and makes DS a viable alternative to the other certainty handling methods. In the same paper, the authors also point out that when this restricted form of DS is examined it becomes clear

that the early CF system of Shortliffe and Buchanan is in fact a special case of the reduced DS system.

2.2.4.4 Fuzzy Set Theory

Zadeh idependently proposed that the use of fuzzy set theory could facilitate the representation of the inherent vagueness of many linguistic terms used in common discourse. Examples of such a term might be 'tall' or 'young' where the term has no absolute meaning in terms of a numerical measurement of either height or age respectively. Using fuzzy set theory, or possibility theory, it is possible to take a term like 'tall' and produce a set membership function which is able to give some measure of belief that a particular instance is a member of the set of tall. Dubois and Prade (1985) have since described in great detail the use of possibility theory and set membership functions in assigning measures of belief to categorization terms for numeric variables. For example we might use the term 'predominant' to describe a mineral which accounts for greater than 80% of a rock mode. However, the 80% boundary is actually completely arbitrary since if we had a rule about quartzite which required quartz to be predominant and quartz was 75% we would still have a high belief that the rock was a quartzite and therefore logically that quartz was predominant. Unfortunately fuzzy sets do not appear to be of any great value in inference, and as suggested by Batnagar and Kanal (1986), are more useful for representing vague set membership for linguistic terms than for representing and handling uncertainty in world models. However, some workers have attempted to make use of fuzzy reasoning in inference systems (Applebaum & Ruspini 1985).

2.3 IKBS and Geology - a Historical Link.

The domain of geology is one which has presented IKBS researchers with interesting problems for many years, although this has resulted in only a few working systems. The first link between IKBS research and the geologist occurred with PROSPECTOR (Duda *et al.* 1978). This system was designed for use by the United States Geological Survey in assessing the viability of a variety of mineral prospects, the value of the system being its ability to advise the geologist on where best to do exploratory drilling. Correct advice at this stage obviously saves a large amount of money by avoiding drilling costs at badly chosen sites. PROSPECTOR requires the user to input field data for the area in question and produces a map as output giving the areas of most interest for future exploration drilling. The system works on the basis of a set of models for the major environmental settings for mineral deposits (Duda *et al.* 1979), many of which have been tested for performance against the expert (Gaschnig 1982).

Once a model has been chosen by the user, PROSPECTOR attempts to apply the expert knowledge to the data about the area of interest. PROSPECTOR asks the user for input of data as it is required but unfortunately requires all of its questions to be satisfied rather than only those of importance. PROSPECTOR then applies a system of Bayesian probability (Duda *et al.* 1978, 1979) to the data using its production rules and finally comes up with its advice on the potential of the prospect. The only widely reported success story of the PROSPECTOR system has been its discovery of a large, unexpected molybdenum deposit adjacent to an actively worked prospect (Campbell *et al.* 1982).

Following the lead of the MYCIN system for medicine, which was stripped of its knowledge to form the domain independent EMYCIN inference engine (van Melle 1979, van Melle *et al.* 1984), such domain independent inference engines, now referred to as shells, have found widespread use as tools for knowledge engineering. PROSPECTOR was also stripped of its geologically specific knowledge to form another of the early shells known as KAS. Thus indirectly geology has been responsible not only for one of the

earliest successful 'expert systems', but also for one of the earliest expert system building tools.

Since PROSPECTOR, only one other geological system appears to have had such success in the both the research literature and in the commercial environment. DIPMETER ADVISOR (Davis *et al.* 1981, Gershman 1982, Bonnet & Dahan 1983, Smith & Baker 1983) represents one of the first major in-house developments in the expert system world (Baker 1985). The system has been developed by Slumberger Doll research at Palo Alto for use in well log data interpretation. DIPMETER, as the name implies, is used to interpret the logs returned by the dipmeter tool at wellsite. This tends to be a very expert task as patterns of dips have to be recognized and regional dip effects allowed for when attempting to infer structures in the subsurface geology. In the case of DIPMETER the IKBS section of the system forms only about 15% of the overall system, the other 85% being used to carry out recalculation and drive the graphic user interface. The system enjoys reports of great success in the Californian basin for which its knowledge base has been built; unfortunately when it was deployed in the North Sea the system demonstrated a high failure rate and has been reduced to the task of graphic representation only with no real AI component being used (Slumberger, pers. comm.). Since the development of DIPMETER, Slumberger have developed further log interpretation systems which are capable of deriving the sedimentary environment and facies at the time of deposition (Yan 1986).

Classification systems for geologists tend to be smaller systems and are usually developed to suit micro-computers. One such system for fossil identification has been developed on a micro using Prolog (Alexander 1985). This system uses a systematic classification which can be modelled by producing cards, each representing a feature of the fossils, with a grid representing different classifications of fossils. If a fossil shows the feature then a hole is cut in the card at the appropriate grid position. As features are found the cards are stacked together, and any holes left through the complete stack represent the fossil 'classes' which are still possible. This has been represented in terms of set

memberships so that only those fossils which contain all of the features found continue to be considered by the system. Any fossil which does not list a feature is considered as never possessing that feature. This system also shows the interesting feature of being able to quiz the student user, thus testing his/her ability to identify fossils accurately.

Alexander has only developed the system to cope with a geographically and stratigraphically restricted subset of the fossil record. This restricted set however shows the applicability and usefulness of the system, which if the correct control were provided could potentially be developed to cover a wider section of the fossil record. This could be done through the utilization of the natural hierarchy of the fossil classification system, thus first deciding between families before loading the different groups within the chosen family. Once a group is chosen the genera can be loaded then finally the species within the genera.

Another micro based expert system in geology has been developed for the identification of the igneous rocks from hand specimen descriptions. The system, INTAL (Hawkes 1985), is able to cope with user input in a free English notation and uses key words to understand and gather data from this input. Unfortunately this system does not conform to the ideal architecture as described in Section 2.2.1 in that the rules with which the system operates are hard coded into the interpreter rather than forming a separately defined rule-base. This however does not detract from the system performing with a great deal of success in providing accurate classifications, in spite of the memory limitations of the micro. The basic interpreter design of INTAL has also been applied to metamorphic rocks, along with attempts to consider the overall task of rock classification covering sedimentary, igneous and metamorphic (Hawkes, pers. comm.).

Mineral classification has been tackled using IKBS techniques (West 1985). This particular system, developed in Prolog, has been designed to identify minerals in thin section using data easily collected by the user via standard petrographic microscope techniques. Development aims here were to provide a system which could function both as

a teaching tool and as a consultancy tool, as with Alexander's FOSSIL system. West has incorporated some interesting features at the user interface, for handling the differences in meaning expressed from one person to another, in the more subjective, descriptive terms. The most important area for West's system is in the treatment of colour data. For example suppose the expert has written a rule for some mineral which requires the colour to be pink. If the user tells the system that the mineral is red then on a purely true/false basis the mineral is not pink. However, red is really a 'dark pink', or pink is a 'pale red', so the system must be able to represent the fact that two colours are close to each other and that if the mineral is red there is some probability that it is also pink. West's system unfortunately does not appear to meet all of the requirements of a 'normal' expert system as the knowledge is coded in the form of a standard decision table rather than as rules. It should be noted that this is also true of the fossil system of Alexander.

A further area of the geological domain in which expert systems may find application is in geophysical interpretation. Although the author is not aware of any important systems being researched in this area the possibility has certainly been investigated (Denham 1985).

CHAPTER 3

Restatement of Problem

3.1 Classification in Metamorphic Rocks

Many geologists find that metamorphic rocks are often very difficult to classify, due to the fact that they have such complex histories. It is the very nature of metamorphic rocks that makes them so complex, i.e. the fact that they have been 'changed' from a previous rock type to their current form. This means that there are many features of the rocks which might be used in classification. In general there are four basic systems to be considered when classifying metamorphic rocks:

1 General classification based on the composition and/or fabric of the rock.

2 Classification of precursor rock type in terms of sedimentary and igneous classes.

3 Classification of metamorphic environment.

4 Classification of metamorphic facies.

This multi-system classification means that the novice requires regularly to refer to text books or to an expert in order to fulfil all four systems since the knowledge required to make a correct classification is more extensive in many ways than that required for the divisions of sedimentary and igneous rocks. It is often very difficult to know what to look for in reaching a conclusion about all of the classifications. The expert normally has such a vast experience that he/she knows exactly what data is relevant and is also able to look for associated data where a direct piece of evidence is missing or ambiguous, often, for example, inferring the presence or absence of one mineral based on the presence, or indeed absence, of another.

3.1.1 Value of a Computer Classification System in Metamorphics

Since it is possible to create a computer system for classification in other domains, and so assist the novice in attaching the correct names to whatever is being classified, it seems that such a system in the domain of metamorphic rocks might be invaluable. Many systems for classification tend to be coded in languages such as BASIC, PASCAL, C, and FORTRAN, with the decision tree for reaching conclusions being hard wired into the code, see Figure 3.1. Such an approach would almost certainly work in the case of metamorphic rocks but would tend to be fairly inflexible and as with all such systems would be difficult to alter and update. At the current time in metamorphic classification, the ability to easily update the classification scheme used in such a system is extremely important, as a commission is currently attempting to reach a recommendation on how such a scheme should be defined. Obviously if we are to go to the trouble of building a classification system, we want one which carries out its classification in a way which agrees with the standard being used by experts in the field. Two alternatives therefore exist: either wait until a standard is agreed or build a system which can be easily modified and thus keep up with the trends in the field. It is also possible that such a system could, at least potentially, then be used in testing the relative performance of any new classification scheme under consideration.

```
if  fabric = "foliated" then
     if  foliation = "fissility"  then
        if   grainsize = "fine"  then
             rock_name := "slate";
        elseif  grainsize = "medium "  or grainsize = "coarse"  then
             rock_name := "schist";
        endif
     elseif  foliation = "banding"  then
         rock_name := "gneiss" ;
elseif.....
```

Figure 3.1 Example section of a procedural algorithm which 'hard wires' classification knowledge. Such code is inherently difficult to update since additional classes may require complete changes to the logic.

3.1.2 Choice of Expert System Methodology as a Solution

In light of the facts that firstly, the domain of metamorphic petrography is one in which rock classification is a very expert task if done correctly, and secondly, the domain is quite poorly defined and still open to alterations of definitions, the best method for tackling the problem is through the use of so-called expert system, or IKBS, technology (see Section 2.2). Such systems permit the coding of expert knowledge, often only gained by experience, in such a manner that it is relatively simple to alter and update. However, it should be carefully noted that although it is simple, in such declarative systems, to add knowledge, normally in the form of rules, without having to alter the logic of the code, certain new problems do arise in the maintenance of large knowledge bases (see Section 2.2.3). Although major maintenance problems do exist, the ability to simply add, or alter, a rule, or piece of information, to a knowledge base and allow the interpreter program to handle its interactions is extremely useful and appealing. In the domain under consideration this feature simply allows us to change the rules of classification as required when a standard is eventually agreed.

3.2 System Design Considerations for the Metamorphic Domain

Several features of the domain of metamorphic petrography must be considered carefully when planning the design of an expert system, particularly as the system will be targetted at the undergraduate student, and should therefore be able to transfer some knowledge to the user. The features which are perhaps most important are the multiple classification schemes used by the petrographer along with a vast background of information about the hierarchical relationships of minerals and their approximate chemical compositions. Also of extreme importance will be the interaction of the system and the user, both in terms of training the novice what data is to be looked for and also giving textbook type information about the rock classes chosen for a specimen.

3.2.1 Handling Multiple Classification Tasks

As described previously, in Section 2.1.1.1, the metamorphic petrologist is interested in classifying rocks according to more than one scheme. We must therefore deal with the schemes that the human expert uses in our expert system. This could be achieved by using a series of domain specific meta-rules similar to those shown in Figure 3.2 and described in Section 2.2.3.

IF general classification not yet done
THEN consider general classification rules
AND general classification has been done

IF general classification has been done
AND precursor classificationm not yet done
THEN consider precursor classification rules
AND precursor classification has been done

Figure 3.2 Example showing the use of meta-rules in controlling inference in the Metamorphic petrography domain. These rules could potentially schedule the two different tasks, 'general class' and 'precursor class', associated with metamorphic classification.

This type of rule based control would be perfectly adequate for our needs and would also be totally declarative. However, as Aikins has pointed out (Aikins 1980, 1984) the use of a mixed representation where frames represent the control knowledge of the system allows for clear distinction between control information and inference knowledge within the rulebase. By using a similar approach the system for metamorphic rocks will be designed to have frames where the slots are used to represent the classification tasks and the ordering of the tasks will be implicit in the slot order. The expert will be able to allow for changes to this control to occur, whenever the data merits such a change, by using demons (rules which constantly monitor the data and fire at appropriate times to cause instant changes to system strategies), as meta-rules to alter the frame agenda. This agenda will have to be local to a frame so that control within any context will be locally executed.

3.2.2 Modular Hierarchical Knowledge Structure

The frames which will represent control information will be valuable to the expert since they will allow classification to proceed in a stepwise manner with each frame task producing a classification at a particular level leading to the execution of a new frame to carry on with the classification to a more precise level. For example, the system will be able to allow the expert to code a classification which in the first instance deals with differentiating schists, gneisses, hornfelses, slates and quartzites say. Once one of these has been chosen the system can then check if there is a further level of classification within the selected group as shown in Figure 3.3.

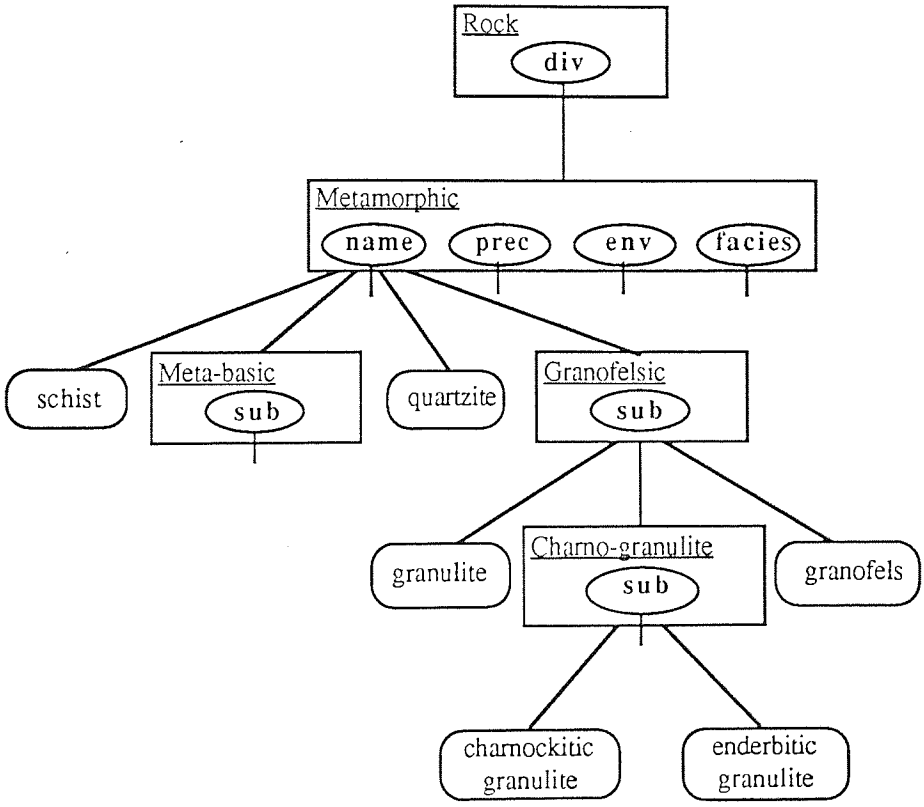

Figure 3.3 Schematic representation of the structure of part of the problem space in metamorphic classification. Contexts are shown in rectangles, leaf nodes in rounded rectangles and tasks in ellipses.

This will serve two important purposes. Firstly the system will be completely modular, in terms of its knowledge base, and will allow the expert to encode the knowledge in a logical stepwise manner, testing system performance at each stage.

Secondly, because the system will be working through a classification which is increasing in detail hierarchically, the user should always be given some sort of answer. For example in the case of a gneiss, the system may not be able to get a satisfactory result in the subclasses of gneiss but will at least report that the rock is a gneiss.

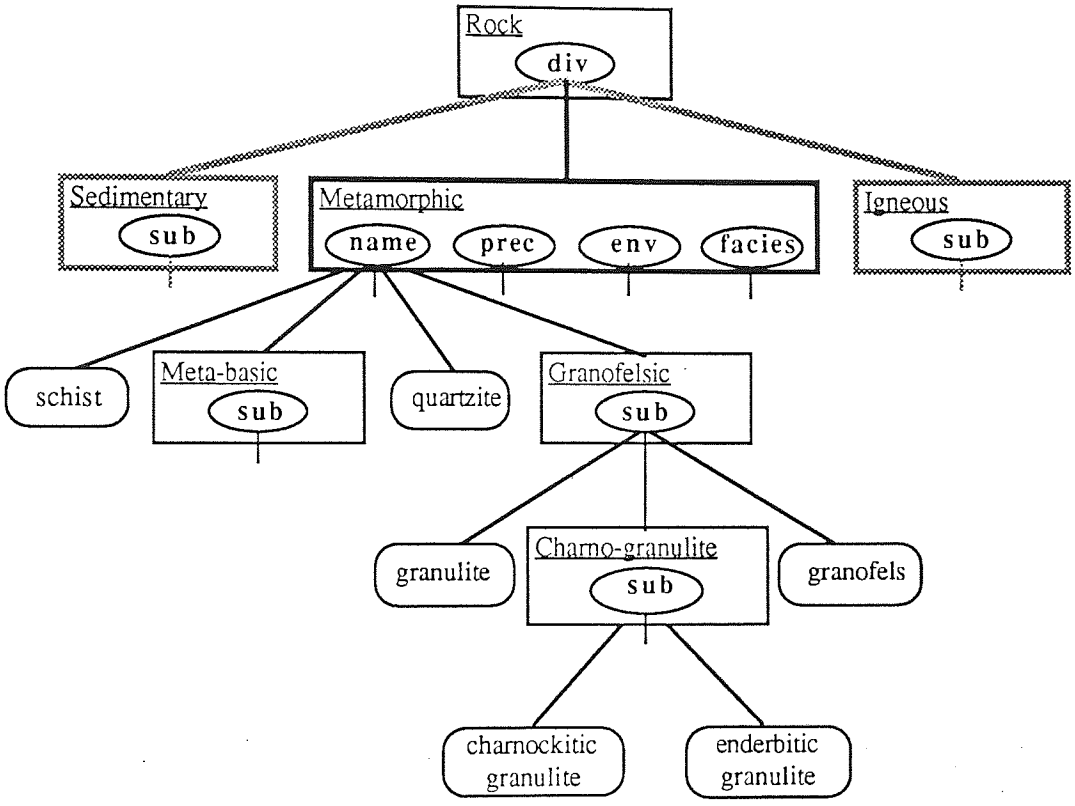

Figure 3.4 Schematic showing how METEX could be expanded to cover the overall rock classification space. The sedimentary and igneous contexts are regarded here as simple add on modules.

A further feature of such a system is its ability to be extended. For the purposes of this project the system will be designed with metamorphic classification in mind. However, should further development be carried out it should be a relatively simple task to add classifications for igneous and sedimentary rocks. To this end the system will be built with the topmost context being 'ROCK'. Beneath the rock frame only the metamorphic division will be encoded and the system will default to this. Should future development be done then the knowledge engineer responsible for this development will only require to

insert the rules to choose between the three major rock divisions (which it should be pointed out is not a simple task) and also the appropriate sub-frames to control the classification of the igneous and sedimentary rocks (see Figure 3.4).

3.2.3 Background Semantic Information

A further feature of the domain important in expert system design is that any geologist caries a vast amount of data relating various concepts, often from geological subdomains not directly related to the domain of interest. The important information from the point of view of building a system for metamorphic rocks is the relationships of mineral names, both hierarchically and also chemically. This is important since in writing the rules for classification the expert will perhaps be interested in whether say amphibole is present or whether an FeMg silicate is present. If the user says that hornblende is in the mode, he would expect the system to know that this means both amphibole and FeMg silicate are present. It is proposed that this type of information will be encoded in a simple semantic net type structure where the expert or knowledge engineer will be required to write the functions required for traversal of the net. The definitions for this net will be in a form which is completely declarative with the system working out what the overall structure is and handling the propagation of new data.

3.2.4 Simple and Logical User Front End

Targeting the system for the undergraduate user requires that the system have a fairly clear and logical user interface. This interface itself will be simple, since it is not a key issue in this project. The control of user interaction is, however. The system will be required to ask the user for data in a sensible order, preferably "teaching" the novice in the domain to look for data in a particular order, i.e. one which is useful in describing rock samples. In order to do this there will be two levels of question control in the system. Firstly, the questions asked before a context can be investigated will be declared by the

expert and inserted as 'preactions' to a task slot. In this way the expert will be able to force particular data to be collected. It will be possible here to have further questions considered dependent upon the answers to one or more previous questions. For example when the system asks if the rock is foliated then dependent upon the response here, a question could be considered to discover the nature of the foliation. In this way initial, important data can be gathered in a logical and 'apparently' intelligent manner, with a clear contextual basis for the questions.

The second level of questioning will be carried out by the inference procedures of the system. This section of the interface will be capable of selecting questions, to ask the user, on the basis of the value of the resulting information in terms of solving the goals of the system. By using this facility the expert will be able to have the system ask particular questions of the user only if the question is likely to resolve some area of doubt not solved by the normally collected data.

3.2.5 Explanation and Text Based Information

As with all IKBSs the system described here will require to give some justification for its decisions. This has been found necessary in other system's simply to give the user the ability to accept or reject the systems reasoning. However, with the targeted user being a novice geologist it seems that it would be very useful to give the system some extra information which it could give on request. For example, if the system decides that some specimen is a 'charnockitic granulite' then it would be very useful to the user if he could ask for further information. This information will take the form of canned text and could be either an explanation of the term used or of the basic factors important in the rock. The text could also however give suitable references to the literature available on the rock as well as perhaps sample numbers for laboratory examples of the same rock for comparison. In this way the system will be able to assist the user not only in identification but also in learning more about a specimen albeit by simply pointing him/her to the correct literature.

This should help to take some of the load off the expert in a situation where he/she might have people simply requiring to find where to look for information.

3.3 Choosing an Expert System Development Tool

Expert system technology has become a fairly important sector of commercial computing with a wide variety of tools for building such systems. These tools are available in three different levels:

1 So-called shells which allow the system builder to concentrate on coding the knowledge for use in the ready built interpreter.

2 High level development environments which provide the basic building blocks for developing more customized interpreters.

3 AI oriented languages which provide features useful in designing AI software, but do not provide the high level building blocks for expert system interpreters.

3.3.1 Expert System Shells

Shell systems undoubtedly provide the simplest and fastest route to building and implementing an expert system. However, many people appear to find that although they can build small prototype systems within a reasonable time scale, development of large systems often becomes very difficult or even impossible. Such situations occur where the detailed knowledge used in a particular domain does not truly match the representation provided by the shell, which will often have been developed by stripping the knowledge from a previous system, leaving only the interpreter. Instead of concentrating on extracting knowledge from the expert, the knowledge engineer soon finds that, in such situations, he/she is spending a great deal of time 'bending' the knowledge base to fit the shell

representation. This problem is unlikely to be resolved until a shell is developed with a representation which is truly domain independent, a feature which will only be obtained when we are more fully able to understand and model how humans represent and use their knowledge.

3.3.2 High Level AI Programming Environments

In light of the danger of beginning work using a shell, and risking the discovery that the knowledge to be represented will not fit the shell, the next best tool would seem to be the high level expert system building environments. Such environments allow the knowledge engineer to build an interpreter which best meets the needs of the domain in which he is interested. The building blocks for doing this are normally provided, with uncertainty methods, problem solving strategies, representation schemes and user interface modules being built together as required. In this case, when the knowledge engineer finds that the knowledge doesn't quite match the interpreter, it is possible, rather than altering the knowledge, to alter the interpreter. The only problem encountered with such systems is the prohibitive cost, both of the software and the hardware normally required to run it. This is of course not true of all of the high level environments, but is certainly true of most.

3.3.3 Low Level AI Languages and Toolkits

The last set of tools available for building expert systems are the languages often referred to as AI languages. These tend to be languages which have been designed with an ability to handle text and easily associate symbols. The two main contenders in this category are Lisp and Prolog. Both languages have been used in the construction of research systems and each offers the system builder a different set of useful features, although neither provides a wide variety of ready built tools to speed development. A third very valuable language, or rather environment, is available in the form of the POPLOG

package. POPLOG provides the programmer with an environment in which not only is the language (POP11) suitable for AI use but also many example programs and basic building blocks are provided. The building blocks provided however tend not to be very sophisticated and require some work by the programmer if they are to be used as part of a larger system. This appears to provide a low cost solution to the problem of choosing a tool for building an expert system while ensuring that the tool itself is unlikely to become restrictive of the knowledge which can be coded. In fact the POPLOG environment is in many ways the most flexible tool to use since not only does it provide its major language POP11 but also provides a version of each of Lisp and Prolog along with access methods to any other language available on the system being used. A further valuable feature of the POPLOG package is that it is available for the DEC VAX range of computers running VMS as well as the more specialized AI workstations. This means that any system developed in the environment should be portable, and also be made available to the end user easily. In the case of this project where the end user is the student in a lab, all that is required for access to the metamorphic expert system (METEX) is a suitable terminal connected to the University central computer.

3.4 Plan of System Development

The production of an expert system for the domain of metamorphic petrography, following the design considerations discussed above, can be described in four stages:

1. Firstly a simple expert system tool was chosen for experimentation and to clarify requirements in an inference interpreter. The chosen system, for this first stage, and its further development are discussed in this chapter.

2. The second stage in the system development involved the production of a control architecture capable of using the inference interpreter whilst being able to handle the complexities of the metamorphic classification problem. The control system designed for this purpose is described in Chapter 5.

3. The third stage of system development was to test the operation of the interpreter within the metamorphic petrography domain. This was achieved by developing a reasonably sized knowledge base covering a subset of the metamorphic rocks. This knowledge base is discused in Chapter 6.

4. The final stage in the project was the evaluation of the completed system considering both performance and also applicability of the chosen technique. The results of this stage are discussed in Chapter 7.

CHAPTER 4

Inference System Design

The inference system of METEX is based on the POPLOG teaching system. Eshell, written by Chris Mellish. The Eshell program is documented as Eprospect in the Ramsey & Barrett (1987) POP11 text book. Although the basic features of Eshell still exist in METEX, the program has gone through several stages of development. It is intended therefore to only briefly outline the important features of the original program, and then detail the major alterations which have been implemented.

4.1 Original Eshell Program

In its original form Eshell was designed as a dependency graph of propositional clauses as shown in Figure 4.1.

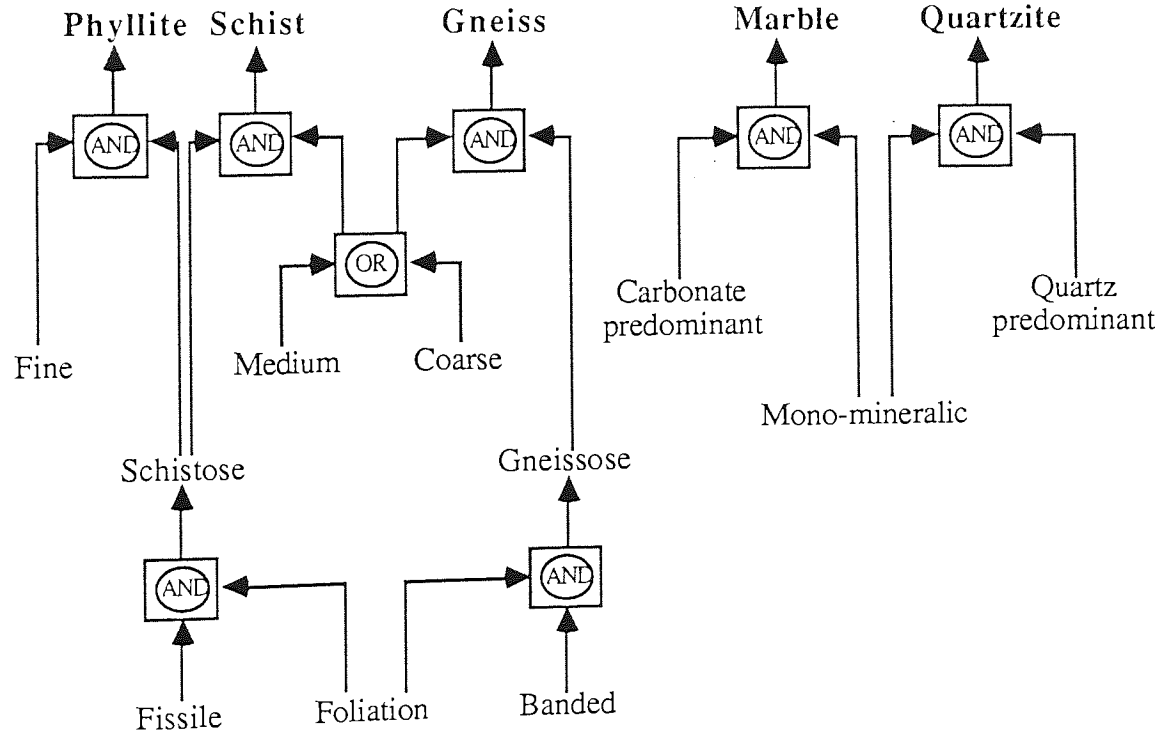

Figure 4.1 Simple dependency graph representing the distinction between phyllite, schist, gneiss, marble and quartzite.

(a.)

name <node name>
lowerbound <lower bound on plausibility>
upperbound <upper bound on plausibility>
expression <conditional expression>
parents <list of nodes dependent upon this node>
fixed <status of the node's measure of belief>

(b.)

name schistose fabric
lowerbound 0.0
upperbound 1.0
expression and<foliated
fissile>
parents [schist phyllite]
fixed false

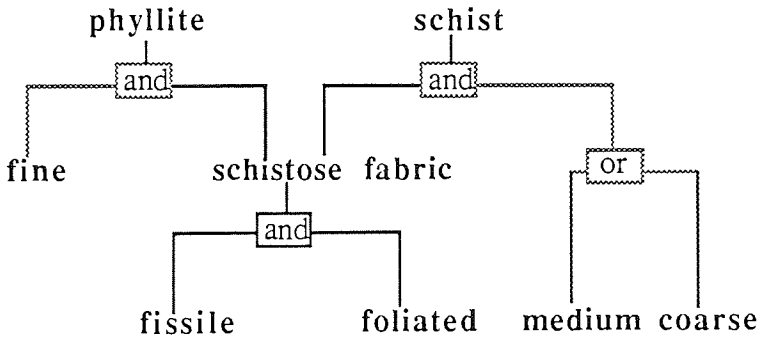

Figure 4.2 a) Shows the general structure used in a node definition within Eshell. b) Shows an example node definition from Eshell detailing the construction of that node into a portion of a dependency graph.

Each proposition in the graph is similar to the nodes of the IRIS system in that it has a frame-like table of data attached to it as shown in the example node in Figure 4.2a. The most important entries are those which define the other nodes to which any one proposition is linked. Two such entries exist in Eshell nodes as shown in Figure 4.2b. The first is PARENTS which stores a list of all of a node's parents, that is all of the propositions for which it forms part of the evidence. Each proposition also stores a logical expression which forms the inference rule for that proposition, serving a similar purpose to the conically structured decision tables of IRIS (Trigoboff 1976, Trigoboff &

Kulikowski 1977). By using both of these pieces of information it is possible for Eshell to propagate data by forward chaining through the parents of a propositions, or to prove propositions by backward chaining. In this way Eshell is very similar to the PROSPECTOR program of Duda *et al.* (1978) and is capable of both accepting unprompted input from the user or directing the questioning of the user.

One of the special features of Eshell is the way in which its 'certainty system' is used to direct the system to ask only those questions which might prove beneficial in proving some goal (Mellish 1985). In order to discuss this it is necessary to first discuss how Eshell, in its original form, handles uncertainty. Mellish originally designed Eshell around a probability range system, where the range boundaries have similar meanings to those in Dempster-Shafer theory. Each proposition in the Eshell dependency graph has attached to it a lower and an upper bound on its probability. The lower bound represents the amount of evidence, or support, for the proposition (sA) whilst the upper bound represents the plausibility of the proposition (pA), that is the maximum possible certainty with which the evidence might cause belief in a hypothesis A. Throughout the rest of this work the ranges will be represented as a list of two values [sA pA]. Every proposition begins with a range [0 1] representing a condition where nothing at all is known about the proposition. This range, and its meaning, is represented, along with a few other important ranges, in Figure 4.3 in order to clarify what is being shown by any pair of numbers.

In Eshell the rules for propagating probability ranges are very simple:.

1. Where two propositions are conjoined as an AND clause the conjunctive probability range is found by taking the minimum sA and minimum pA values.

2. Where two propositions are disjoined as an OR clause the disjunctive probability range is calculated by maximizing the sA and pA values.

3. The negation of a proposition is defined as having beliefs s~A = (1-pA) and p~A = (1-sA).

4. In a production rule, the probability range for the premise clause is calculated according to rules 1 to 3 above. The sA and pA values for the conclusion are then calculated by multiplying the premise values by the certainty factor attached to the rule.

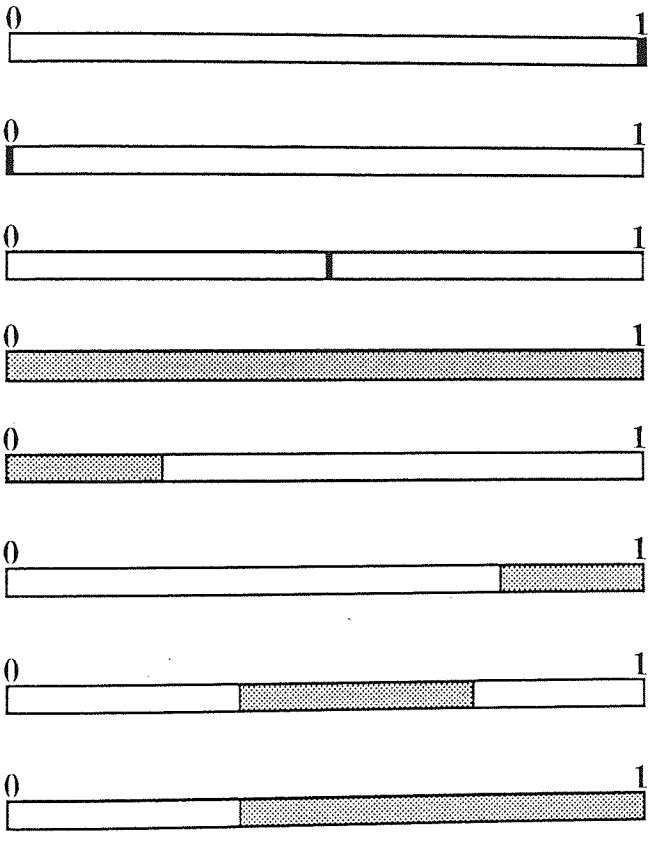

Figure 4.3 Examples of Eshell belief ranges with accompanying interpretations. Shaded areas indicate the range within which the true measure of belief must lie.

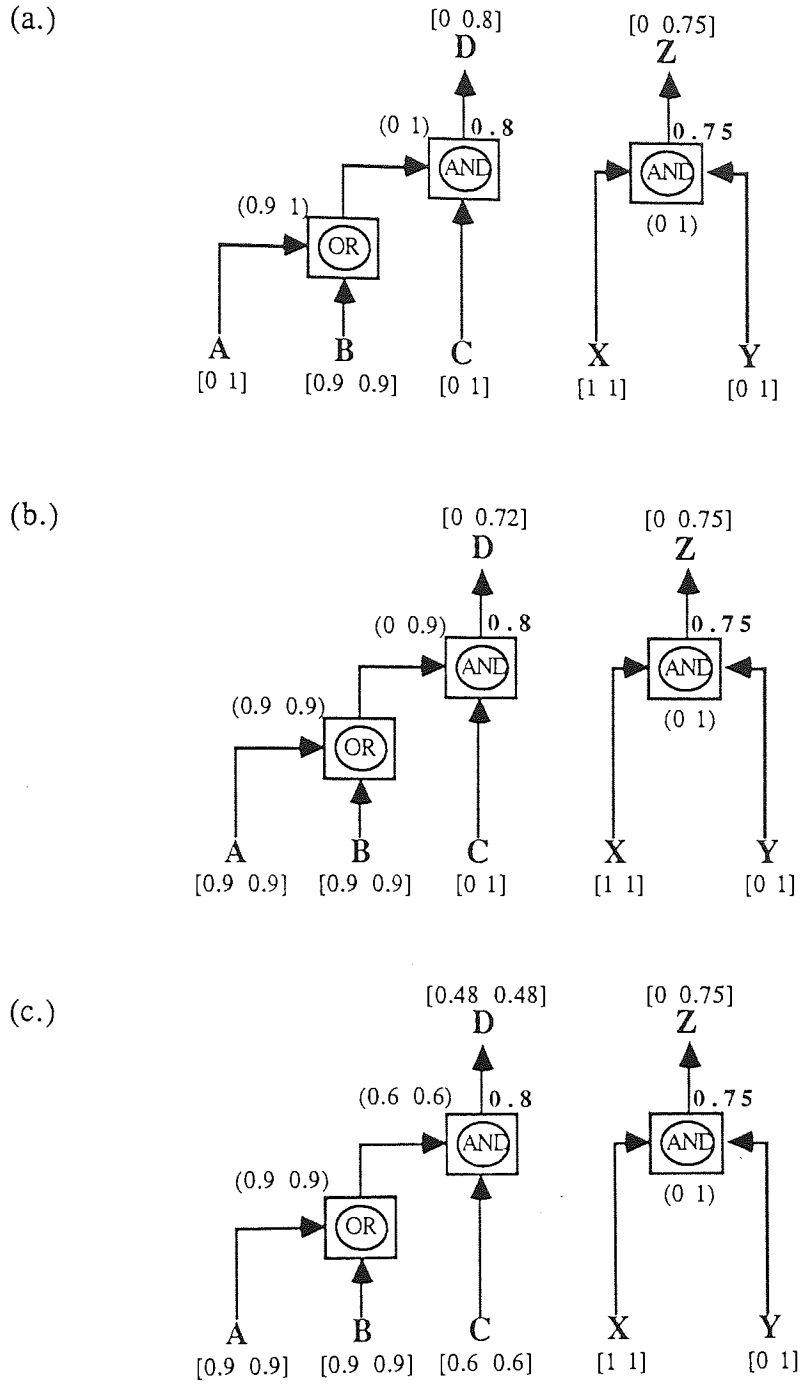

Figure 4.4 Example portion of a dependency graph showing the propagation of uncertainty in Eshell. The graph represents the rules '**IF** (A **OR** B) **AND** C **THEN** 0.8 D' and '**IF** X **AND** Y **THEN** 0.75 C'. Figures in bold represent rule weights, figures in round brackets evidence belief ranges after each combination and figures in square brackets the belief ranges for individual propositions.

The inference rules described above have been used in Figure 4.4a to demonstrate how probability values interact across a small section of a dependency graph in Eshell. It

is clear that if we were to alter our belief in proposition 'A' to be [0.9 0.9] as opposed to [0 1] then the disjoint group of 'A OR B' has a new range of [0.9 0.9]. If 'C' maintains a range of [0 1] then the conjoined group of '(A OR B) AND C' will assume a range of [0 0.9]. multiply this by 0.8 for the rule and we have a resultant range for D of [0 0.72], shown in Figure 4.4b. If we now alter 'C' to have a range [0.6 0.6] then the clause '(A OR B) AND C' now has a range [0.6 0.6] (from [min(0.6,0.9) min(0.6,0.9)]), resulting in 'D' assuming the range [0.48 0.48], shown in Figure 4.4c.

4.1.1 Backward Chaining and Question Selection

A very important feature of Eshell's operation is its ability to search for propositions which are likely to have some desired effect in satisfying a chosen goal while ignoring propositions which have little or no value. The algorithms used to do this are discussed by Mellish (1985) and are based upon alpha-beta graph searching. This is in fact a similar approach to that described by Applebaum & Ruspini (1985) for the ARIES system which uses a hypergraph representation similar to that of Eshell. To demonstrate question selection in this way return to the example dependency graph in Figure 4.4a and assume that proposition 'C' and proposition 'Z' are two competing hypotheses. Since 'D' has as yet not been fully evaluated, it still has a range [0 0.8]. We could adopt one of two different strategies to produce one hypothesis with no competing hypotheses. Hypotheses are taken to be competing if their ranges overlap at all, since this means that either proposition could feasibly move to its range maximum while the competing proposition could move to its range minimum. Some possible pairs of competing hypotheses are shown in Figure 4.5 along with the possible strategies which might be used to resolve the overlap.

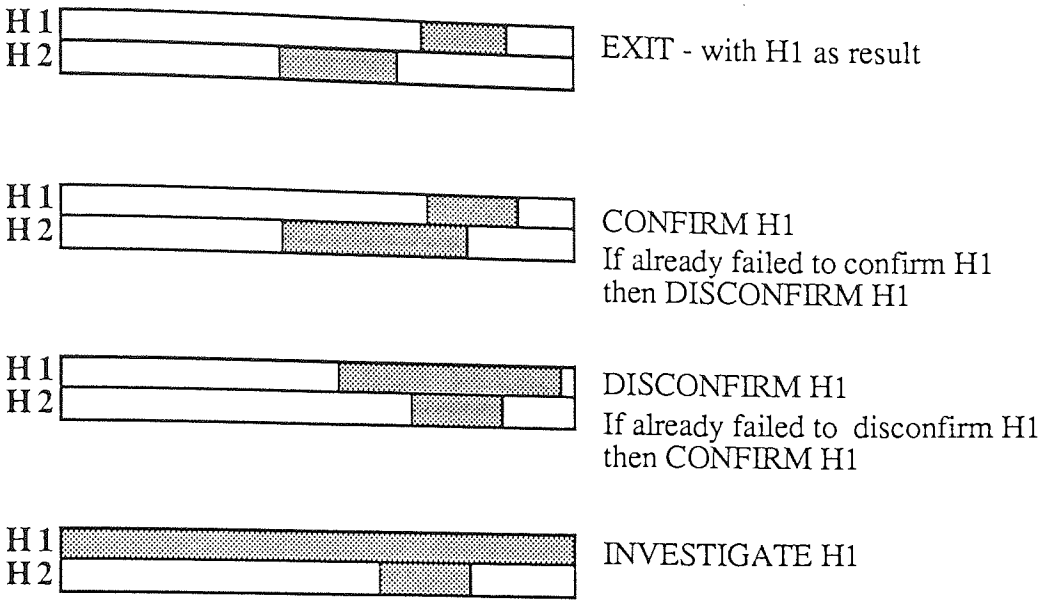

Figure 4.5 Example showing how Eshell chooses its questioning and inference strategy depending upon ranges of belief for competing hypotheses, H1 and H2. The chosen strategy will then be responsible for directing the question selection procedure.

Let us assume, returning to the example in Figure 4.4a, that the system chooses to attempt to confirm the hypothesis 'D'. This can only be achieved if the lower range value of 'D' can be moved, in the first instance, to some value greater than its current value of 0. With this as its aim the system now attempts to find any proposition upon which 'D' is dependant, which might potentially have its lower range value increased, thus causing an increase in the lower range value of 'D'. If we now look at the dependency graph beneath 'D', we find that two different propositions could be evaluated, either 'A' or 'C'. However if we consider, as Eshell does, how the values in a rule are combined, we find that the lower range value of 'D' is being completely controlled by the lower value of 'C'. Even if the range of 'A' could have its lower limit increased, such an increase would be of no value unless 'C' is increased. In light of this the system now only asks the user about 'C'.

If the user gives 'C' a range of [1 1] then 'D' becomes [0.72 0.8]. Now if the system decided to increase the lower limit on 'D' above 0.72 then 'A' would have to be asked. However, if 'C' were given a range of [0.6 0.6] by the user then 'D' would gain a

range of [0.48 0.48]. In such a case there is no value in asking about 'A' since its evaluation will have no real effect on the outcome of inference, an effect shown in Figure 4.4c.

4.2 Deficiencies and Alterations made to the Eshell System

In its original form, Eshell is only really intended as a demonstration system. In order to make it useful for tackling realistic metamorphic problems several alterations have been implemented to extend the inference power of the system. The main alterations made were: the introduction of procedural attachments to propositions, allowing variables to be used in evaluation; the addition of a pattern matching facility to allow the expert to encode templates for generalized propositions; and also a reworking of the MYCIN type certainty factors (Shortliffe 1976, 1984, Shortliffe & Buchanan 1975), to fit the Eshell system allowing consistent negative inference and the combination of evidence. These alterations to Mellish's original program have worked together to produce a powerful and flexible inference tool which is more capable of handling real problems.

4.2.1 Variable Access in Propositions

Due to the nature of the inference net in Eshell, where every node is a clearly defined proposition, it is not possible to use the values of variables in inference. Such a feature is required in order to allow the results of questions asked by the questioning routine, described in Section 5.2.3, to be used by Eshell whenever it is activated with a ruleset. In order to allow this type of variable access, a facility has been implemented which allows a procedural attachment to be linked to a proposition in place of a rule. Every proposition still requires to be defined to the system, but in cases where the belief in a proposition can be found from some variable, then that proposition is simply a categorization of the variable. A very simple example might be where the questioning routine finds that the variable, grainsize, should have a value "medium". Now if we allow the proposition 'the

rock is coarse grained' to have its measure of belief found from the result of the procedural statement, grainsize = "coarse", the system can successfully realize that the proposition should not be believed, that is in Eshell it should have a range of [0 0].

The nature of this procedural attachment facility allows for a great deal of flexibility in defining propositions. For example, if we were to ask the user to give the variable 'grainsize' some numerical value, then the individual grainsize categories could be tested for by the associated propositions, each category being defined as a simple comparison of the 'grainsize' value to the numerical range for that category. This can prove very useful where there is some doubt, or argument, about how to categorize some feature, potentially leading to some confusion as to what an individual actually means in using a term. If the feature can be given a precise value then the system can be allowed to do the categorization, ensuring the meanings attached by the expert to terms within the system are consistent with those used in the categorization of input.

4.2.2 Patterns in Propositions

In geology as in many other domains, the expert makes use of many very similar terms. For example, we might make the statement that 'the rock is biotite bearing' based on the presence of biotite in the mode. Since Eshell uses a proposition-based dependency graph, all of the propositions within that graph must be defined. (This is not strictly true since if a proposition is not defined by the expert, Eshell assumes that it is at liberty to ask the user about the proposition. However, in most cases we do not want this to happen, and must therefore define the proposition.) In the example proposition, 'the rock is biotite bearing', we are interested in the proposition that the rock contains a particular mineral in its mode. The method used to find if the rock is biotite bearing is the same as that used to find if the rock is quartz bearing. No matter what the mineral name the method is the same. Since there are some 2000 minerals that the expert might decide he wanted to consider we would potentially require to define 2000 different propositions with 2000 slightly different

rules. If the expert now decides that he might wish to infer that '<any-mineral> is a major component' then we have another 2000 propositions and associated rules.

During the alterations to Eshell this problem was tackled and solved by combining two different solutions. Firstly, Eshell was given the ability to match definite propositions against a general pattern. By using this feature the expert simply defines the template for a proposition like 'the rock is biotite bearing'. The variable terms are picked up in the pattern-matching process so that any rules or expressions within the proposition can be instantiated with the correct terms for the particular proposition. This allows the expert to simply use propositions which match the pattern without worrying about defining them. Instead of automatically asking the user about the proposition Eshell checks to see if it matches any of the definition templates. If a match is found then the definition is added to the dependency graph for use rather than asking the user. It should be noted that this feature may not necessarily prevent the user being asked about the proposition, as the template might only produce some canned text, specific to the proposition, for use in explaining the expert's definition of the proposition.

In the example of propositions relating to minerals, an estimate of the potential number of definable propositions was given. It is unlikely that the expert would ever wish to consider every mineral species in a real system. However, if we assume a worst case where all of the minerals are mentioned in each of say ten different templates, then we have about 20,000 propositions in a dependency net. Such a situation must cause computational problems, especially if each proposition uses some set of daughter propositions for inference, causing a combinatorial explosion type problem. In order to avoid this problem as much as possible, template matching is only allowed to occur in rule sets which have been activated through a call from the main frame interpretation loop, concerned with slot evaluation (see Section 5.2.3).

4.2.3 Revised Certainty Mechanism

The limitations of the inference mechanisms used by the original Eshell program are very apparent when attempts are made to construct a set of rules for some realistic problem. The main difficulty is that Eshell does not allow the expert to implement rules other than those which are completely 'definitional' in nature. For example, suppose we wished to construct a rule which states that: IF it is snowing THEN 1 it is winter. This rule in Eshell would be treated as 'definitional' in that if 'it is snowing' were found to be false then 'it is winter' would be made false. This is not however correct since it may well be winter although it is not snowing. In order to implement this sort of rule correctly it is necessary to have a clear distinction between what constitutes evidence against a proposition and what constitutes evidence for. In order to gain a clear distinction one possible solution is to use a number scale from -1 to 1, as used in MYCIN, to represent both the measure of support for a hypothesis and the weighting of rules. By using the MYCIN type certainty scales it becomes possible to allow a great amount of flexibility in inference.

(1) **IF** clear blue sky
AND sun is shining
AND people are swimming
THEN 0.7 it is summer

(2) **IF** it is snowing
THEN 1.0 it is winter

(3) **IF** it is snowing
THEN -1.0 it is summer

Figure 4.6 Example set of rules demonstrating the use of negative reasoning to encode an exception to a normally correct rule.

The first major benefit from this system is linked to the 'definitional' type rules. In order to implement rules like the snow and winter example above we really want to have two weights on the rule. The first weight reflects the amount of certainty with which snow infers winter while the second weight reflects the amount of certainty that not snow infers

not winter. This second weight is used to make negative inference about the winter proposition. Negative reasoning is a feature which is absent from Eshell, but when implemented correctly becomes a powerful reasoning tool. For example, consider the rules shown in Figure 4.6.

In these rules only single weights are given to indicate the support for the conclusion when the conditions are true; the weights for situations where the conditional clause is false would be 0 in each case, so they have been left out. Rule 1 states that 'when people are swimming and the sun is shining in a clear blue sky then it is probably summer but it could still be summer even if any of these facts are not true'. Rule 2 states that 'if it is snowing then it is definitely winter but if it is not snowing it could still be winter. Rule 3 tells us that 'if it is snowing then it is definitely not summer and also if it is not snowing it is not necessarily summer'. Rule 3 can be used as an exception to Rule 1 since it is possible to have a blue sky with the sun shining and people swimming even though it is winter. Because the weight is so strong in Rule 3 it will overpower the evidence for summer in Rule 1 causing the system to have a strong belief in winter with some certainty that summer can be ruled out. It would of course be possible to simply state that summer and winter are mutually exclusive avoiding the need for Rule 3. However, rules as written above do work and in some ways give a more causal representation of the exclusivity. Also the expert is possibly more likely to remember that a piece of evidence negates some hypothesis rather than remember, or even want to remember, all of the other hypotheses which form an exclusive set. Rules as written in this example can be considered to show that the evidence is sufficient with respect to the conclusion but not necessary.

Certainty Evaluation of Propositions

In moving to a MYCIN type certainty system it has been necessary to alter the way in which Eshell evaluates its propositions. Originally Eshell evaluated individual rules on a minimise-maximise system. This feature has been maintained in evaluating the premise clause of individual rules in METEX. However, Eshell minimised all conjunctive clauses

in such a way that when a single proposition in the premise was found to be untrue, that is had a range of [0 0], then that value was propagated for the rule. Because of this, as described in Section 4.2.2 it was impossible to allow for rules which could tell us something about the truth of a conclusion but nothing about its falsity; instead the only rules which could be implemented were purely definitional in nature. In order to allow for the use of suggestive rules in METEX, it has been necessary to use two weights on the rules. The first weight is for use when the premise clause evaluates to have a positive value, the second weight being used when the premise evaluation is negative. The mini-max system of Eshell also failed to allow for multiple rules to be used in evaluating a proposition, so that it was impossible to have rules with very low weights helping to increase or decrease the certainty of a proposition by combining their individual evaluations. Combination in such cases is carried out in METEX by using the combination functions from MYCIN, which are in fact necessary for the implementation of the double weighted rules just described. Care was taken in this implementation of the combination functions to ensure that the ordering of rules would have no bearing upon their combined value. This has been achieved by ensuring that all of the negative and positive evidence is pooled separately before combining the two. This is very important as the EMYCIN formulae if applied to rules as they appear do not prove to be commutative in their evaluation. For example, consider three rules producing a list of CF values (-1 1 1). If we combine these in the order in which they appear then using Equations 2.3, 2.4 & 2.5 we find that the first two numbers give us 0 which combined with the third gives 1. However if we re-order the numbers to (1 1 -1) then the first two give us 1 which combined with the third gives a value of 0 which is not the same as the 1 value from the previous order of the same rules. If we now alter our approach to calculate the positive and negative values separately we get -1 and 1 combining to 0 every time no matter what the number of rules or their order. Three levels of evaluation are now necessary in calculating the certainty of any proposition. The first level of evaluation is that of the individual rule. Here the rule can be either a simple one weight rule such as shown in Figure 4.7a, or a double weighted

rule showing the weights to be used when the conditional clause evaluates to true or false, as shown in Figure 4.7b.

(a)

IF <conditional clause>
THEN w conclusion

or

IF <conditional clause>
THEN -w conclusion

≡

IF <conditional clause>
THEN w ~conclusion

(b)

IF <conditional clause>
THEN [w1 -w2] <conclusion>

≡

IF <conditional clause>
THEN w1 <conclusion>

and

IF ~<conditional clause>
THEN -w2 <conclusion>

or

IF <conditional clause>
THEN [-w1 w2] <conclusion>

≡

IF <conditional clause>
THEN -w1 <conclusion>

and

IF ~<conditional clause>
THEN w2 <conclusion>

Figure 4.7 Examples showing how the sign of weights in rules alter the logical meanings of the rule. The symbol '~' indicates logical negation of an hypothesis or a proposition.

To evaluate a rule the system must first evaluate its conditional clause. This is done by the same minimise and maximise rules as used in the original version of Eshell described in Section 4.1. Once a value has been obtained for the conditional clause, the sign is checked to discover which rule weight should be used for further calculation. If only one weight is given in a rule, it is assumed that this refers to a positive conclusion to the conditional clause. When two weights are attached to the rule, the first is used in positive conditionals and the second in negative conditionals. The rule for propagation here is simple: if the conditional certainty is positive then multiply the 'positive

conditional' weight by the conditional certainty. If the conditional certainty is negative then the 'negative conditional' weight should be used.

Rule evaluation is carried out according to the rules described above. However, the task is slightly complicated by the use of certainty ranges in Eshell. Although the number system has been made MYCIN-like in meaning, the valuable information made available by having ranges of certainty is maintained. This allows the system to draw distinctions between propositions which have completely uncertain evaluation, i.e. a range [-1 1], and propositions which have been evaluated to have complete uncertainty, a range of [0 0]. A range of [0 0] occurs when:

1 The user does not know the value of a proposition, i.e. it is 50/50 whether the proposition is true or false.

2 The system evaluates conflicting evidence about the proposition putting it into a 50/50 belief in the proposition's truth or falsehood.

3 The system propagates either its own or the user's uncertainty through a production rule.

In order to produce a certainty range in evaluating a rule METEX must be able to handle situations where the lower range boundary is negative and the upper boundary is positive. This is achieved by adding a rule of inference which states that both the positive and negative conditional weights should produce results. However, if the value of the conditional certainty is of the incorrect sign then a value of 0 should result. This produces a system whereby each of the bounds is considered against the weight for the individual rule. Double weighted rules should be considered as being two separate rules to be combined after evaluation. Consider the rule in Figure 4.8.

IF a
THEN [1.0 -1.0] b ≡

(1) IF a
THEN 1.0 b

and

(2) IF ~a
THEN -1.0 b

Figure 4.8 A double weighted rule showing conversion to a pair of rules to handle correctly the distinction between positive and negative evidence. i.e. the distinction between 'sufficient' and 'necessary' evidence. In this rule 'a' is both necessary and sufficient to infer 'b'.

In this example if the conditional 'a' is found to have a range of [-0.8 0.9] we have Rule 1 evaluating to give [0 0.9] and Rule 2 evaluating to [-0.8 0]. These can now be combined to give 'b' a resultant certainty range of [-0.8 0.9], the combination being carried out by using the MYCIN functions for combining the two lower then the two upper bounds to give the composite.

Once the individual rules are evaluated and any double type rules combined, it is possible to combine any multiple rule units to evaluate the overall proposition. This is the top level of the expression slot of a proposition where a list of rules is stored. To combine these rules METEX uses the MYCIN combination functions to produce a composite range. Each boundary is evaluated across the rules by first combining all of the positive evidence followed by all of the negative evidence. Once this is done the two values can be combined to give the overall range boundary.

This method for evaluating the certainty range of a proposition produces a relatively intuitive behaviour of the range boundaries. Consider a simple example, as shown in Figure 4.9, where several rules are combined in evaluating a proposition. Remember that Rule 1 translates to a more English form of 'if there is a clear blue sky, the sun is shining, and people are swimming, then it is likely that it is summer'. Similarly, Rule 2 translates to 'if it is cold then it is not summer, but if it is not cold then it is summer'. These rules might be imagined as applying in an "ideal world" where summers are always warm.

(1) **IF** clear blue sky
AND sun is shining
AND people are swimming
THEN 0.7 it is summer

(2) **IF** it is cold
THEN [-1.0 1.0] it is summer

Figure 4.9 Example rules used in the text to demonstrate the propagation of evidence in the augmented version of Eshell. Note that these rules demonstrate the difference between 'necessary' and 'sufficient' evidence.

Without having any knowledge about the propositions, 'cold', 'blue sky', 'sunshine', and 'people swimming', and therefore 'summer', all propositions should have a CF range of [-1 1]. A run through the evaluation of 'summer' using inference Rules 1 and 2, demonstrates that the reasoning mechanism is consistent with this.

Evaluating Rule 1 first, we evaluate the conditional statement by minimizing across each of the lower and upper bounds, producing a resultant range of [-1 1]. This range can now be propagated through the rule to produce a new inferred range for summer. Each of the limit values is considered in turn, obeying the rules of inference detailed above. The lower -1 value fails to be used in the rule as only a positive evaluation weight is given. A 0 value is therefore returned as the new lower bound. This can be rationalized if the single weight is actually viewed as a pair of weights [0.7 0] and the rule is evaluated as described later for Rule 2. Going on to propagate the upper range boundary we find that this does not contravene the rules of inference. In this case a value of (1 * 0.7) is returned so that the new intermediate range for summer, from Rule 1, becomes [0 0.7].

Going on to evaluate Rule 2 the system finds that there are two weights, and therefore two rules, 2a and 2b, to evaluate. Rule 2a is evaluated first with the CF range of the condition 'cold' as [-1 1]. Propagating this we find that the -1 weight produces a new range of [-1 0] remembering that, since the lower bound is negative it produces a 0 while the upper bound produces -1 from (1*-1). Evaluating the condition 'not cold' of Rule 2b we find again that the new range is [-1 1], from [-(ubound COLD) -(lbound COLD)].

Propagating this range as before we produce a range of [0 1] from rule 2b. These intermediate ranges can now be combined by evaluating the combination of each of the lower and upper bounds in turn. The combination is carried out using the appropriate MYCIN combination functions, as described in detail above, producing an intermediate range of [-1 1] for Rule 2.

Finally the CF range of 'summer' can be calculated by combining the intermediate ranges from both Rules 1 and 2. Where multiple rules exist the combination of the lower bounds from the rules is carried out in the following order. Firstly the negative values are combined by the formula (a + b + (a*b)) to produce a cumulative total. Next the positive values are similarly combined by the formula (a + b - (a*b)). These two values representing the evidence against and evidence for the hypothesis respectively can then be combined by the function (a+b)/1-(min(abs(a),abs(b))). Repeating this process for the upper bounds a combined CF range is calculated giving an evaluation for 'summer', in our example, of [-1 1].

Evaluations for pairs of combining positive and negative evidence sources are shown in Figure 4.10. These evaluations show the consistent manner in which CF ranges move as information becomes available to the system. Readers should particularly note the example in Figure 4.10(e) where the two rules compete equally resulting in the system having a range of [0 0]. This appears intuitive in that all of the evidence for and evidence against conflict directly resulting in an 'unknown' hypothesis, that is a hypothesis about which nothing can be said.

The overall behaviour of this uncertainty mechanism has been tested against that used by Quinlan (1983) in his INFERNO system, which is regarded by many as the state-of-the-art in certainty handling. In order to test the performance of the uncertainty mechanism described here, changes in the measures of belief in evidence were propagated through some rules. The behaviour of the belief ranges on the inferred hypotheses in each system was then compared. This testing revealed that the belief ranges attached to

individual hypotheses, after propagation, were consistent in both systems, with the ranges moving towards similar point values. However, the values themselves were found to be different for each system. This is because the system described here uses a combining function which is able to increase the degree of belief in a hypothesis if more than one line of evidence infers that hypothesis. INFERNO however uses a simple min-max procedure to reach a combined measure of belief in similar circumstances. Because of this difference the two systems can never be expected to produce the same belief values but should however, always show a consistent meaning in their measures of belief.

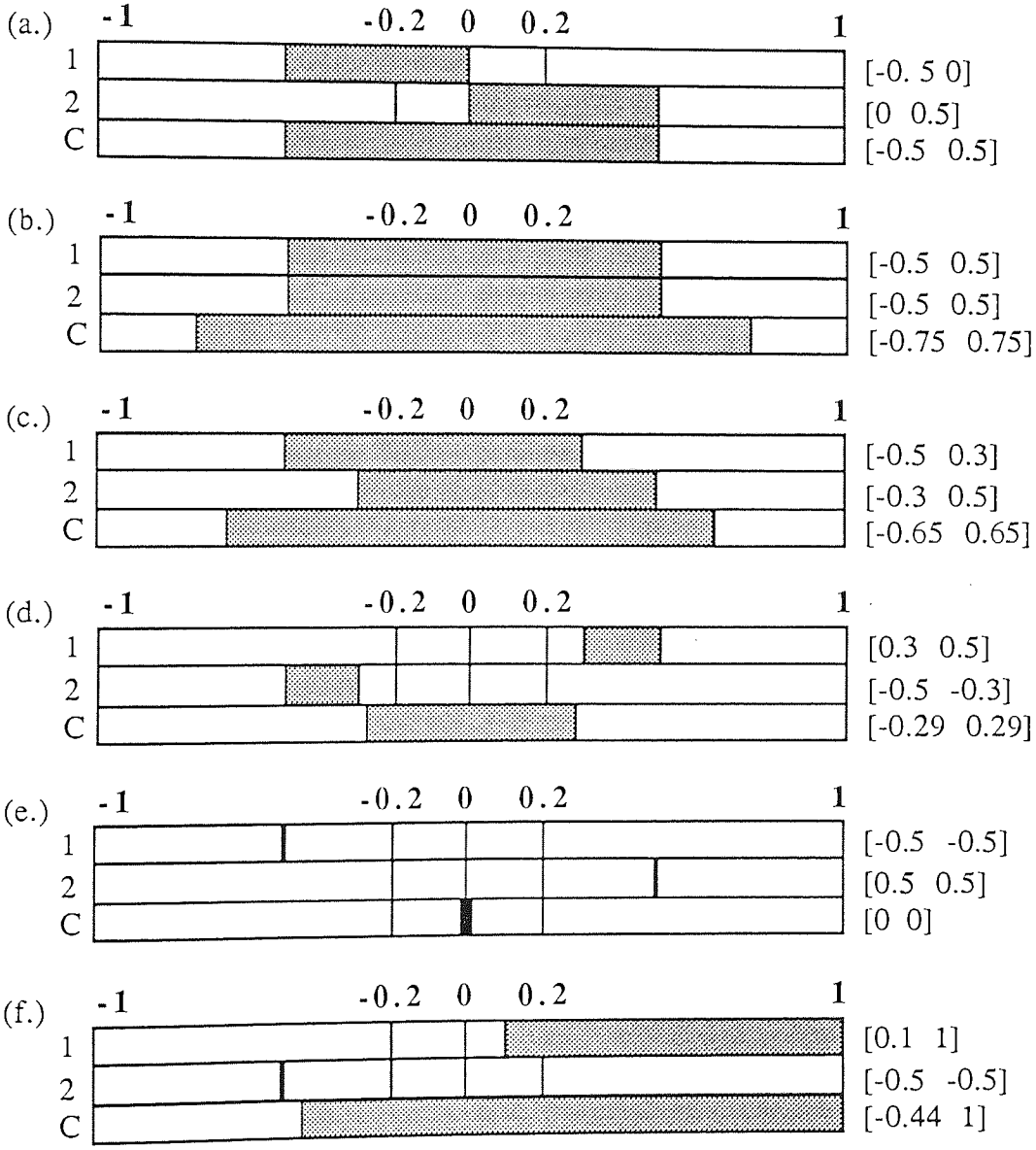

Figure 4.10 Examples of combining rule pairs showing the consistency with which CF ranges are altered as evidence is gathered.

Meaning of -1 to 1 Ranges

In the augmented version of Eshell as used in METEX the range [-1 1] is equivalent in meaning to the original [0 1] range of Eshell. This allows a direct mapping between the two ranges by a conversion function. However, in order to prove the equivalence of the ranges it is necessary to show how an Eshell [0 1] range can be converted theoretically to a [-1 1] range. Recalling that in MYCIN the definition of CF is CF = MB-MD. Also, in Eshell, the range [sP pP] is equivalent to the range [sP (1-s~P)] and also the range [(1-p~P) pP]. This means that each range can provide a set of information, sP which is the minimum support of P, pP which is the maximum support possible for P, s~P which is the minimum denial of P, and p~P which is the maximum possible denial of P. If we now consider sP as the minimum MB of P and s~P as the minimum MD of P, then pP becomes the maximum possible MB and p~P becomes the maximum possible MD. These values can now be combined to form a range within which the CF of P must lie. The lower boundary of CF must come from the maximum MD and minimum MB so that CFl = sP - p~P = sP - (1-sP), and the upper boundary of CF comes from the maximum MB and minimum MD so that CFu = pP - s~P = pP - (1-pP).

This conversion obeys all of the rules about CFs as set out by Shortliffe (1976). As can be seen when a range of [0 1] is converted we get a minimum MB = 0 and a maximum MD = 1 along with a maximum MB = 1 and minimum MD = 0. This produces a CF range of [(0-1) (1-0)] or [-1 1] while obeying the rule that a MB of 1 requires a MD of 0 while MD of 1 requires a MB of 0.

CHAPTER 5

Control Architecture

5.1 Overview of METEX Design

At the simplest level METEX consists of a knowledge base and a knowledge interpreter to make use of the knowledge, as in Figure 5.1. The knowledge base contents are described in detail in Chapter 6, while this chapter along with Chapter 4 discusses the design features and behaviour of the interpreter. In this chapter a description of the control architecture of METEX is given. This control architecture was developed for several reasons. Firstly the domain of metamorphic petrography considers more than one aspect of a rock in classification so that some control method is required which is capable of handling the individual aspects in the order indicated by the expert. Secondly, the classification categories used by the metamorphic petrologist tend to be based on scales which often have no natural breaks. Because of this it is necessary that the system is capable of classifying a sample into some category of rocks based on one feature, but then is able to link across to another category when subclassifying the sample if the evidence supports such a link. For example, if we were classifying a rock which had the correct characteristics to be considered a 'gneiss', that is a gneissose texture, but had a composition close to that of an 'amphibolite', then we might want to be able to transfer from the gneiss group across to the amphibolite group, in order to get the best subclassification of an 'amphibolite_gneiss' or a 'banded amphibolite'. In order to gain this type of flexibility one solution is to have an inference control system as described here, along with a well structured knowledge base. The control system described in this chapter allows for the knowledge base, described in Chapter 6, to be completely modular and also declarative. This means that the expert and the knowledge engineer are able to increase the detail to which classification occurs in a stepwise manner, as they become happy with the performance of the knowledge in each previous step.

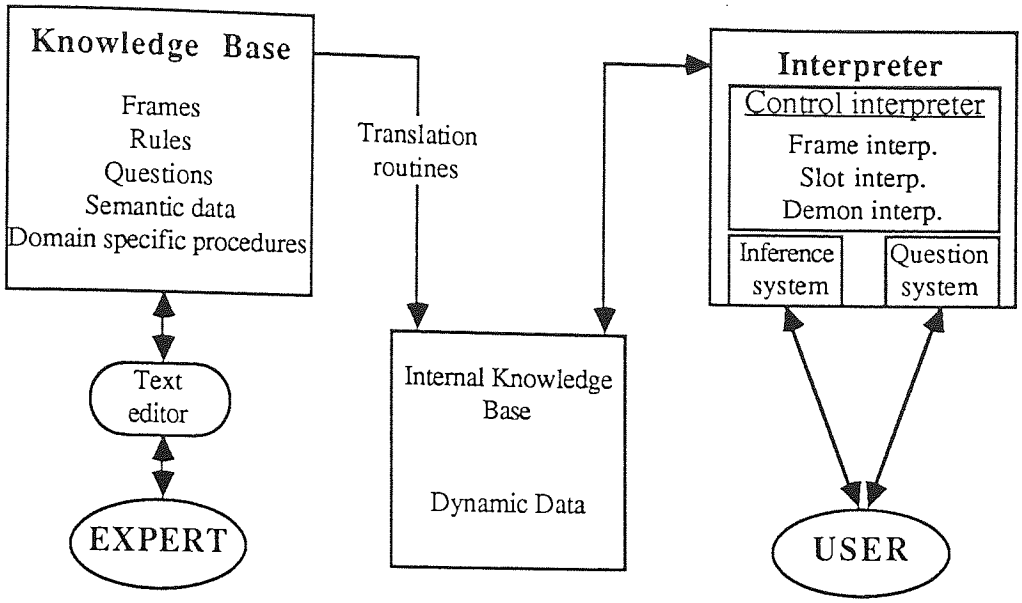

Figure 5.1 Overall architecture of the METEX system showing distinction between knowledge and interpreter.

5.2 METEX Knowledge Representation

In METEX, domain knowledge is stored in frame-like structures which are used to provide control during a consultation. These frames form the nodes in a meaningful search space within which METEX must find its solutions. However the search space used by METEX is slightly more complicated than most games trees, such as those described in introductory AI texts (Bundy 1980, Hunt 1975, Rich 1983, Winston 1984). Where as game trees tend to be simple branching structures, METEX exhibits slightly more complex overlaying of search trees. Because a frame in METEX is able to cause multiple search spaces to be activated then the overall appearance of the entire search space might be viewed as in Figure 5.2, in which the tasks are drawn as elipses, each ellipse representing the root of a new tree.

Apart from the control knowledge, METEX also includes structures for: question definitions, which contain the information necessary for the system to interrogate the user; semantic rules, to allow for the implementation of very simple hierarchical relationships; and inference rules which carry out the classification tasks. All of these major knowledge

structures are stored as declarative items which the interpreter uses, as directed in the control information. The inference rules are subdivided into contextually related groups or rulesets.

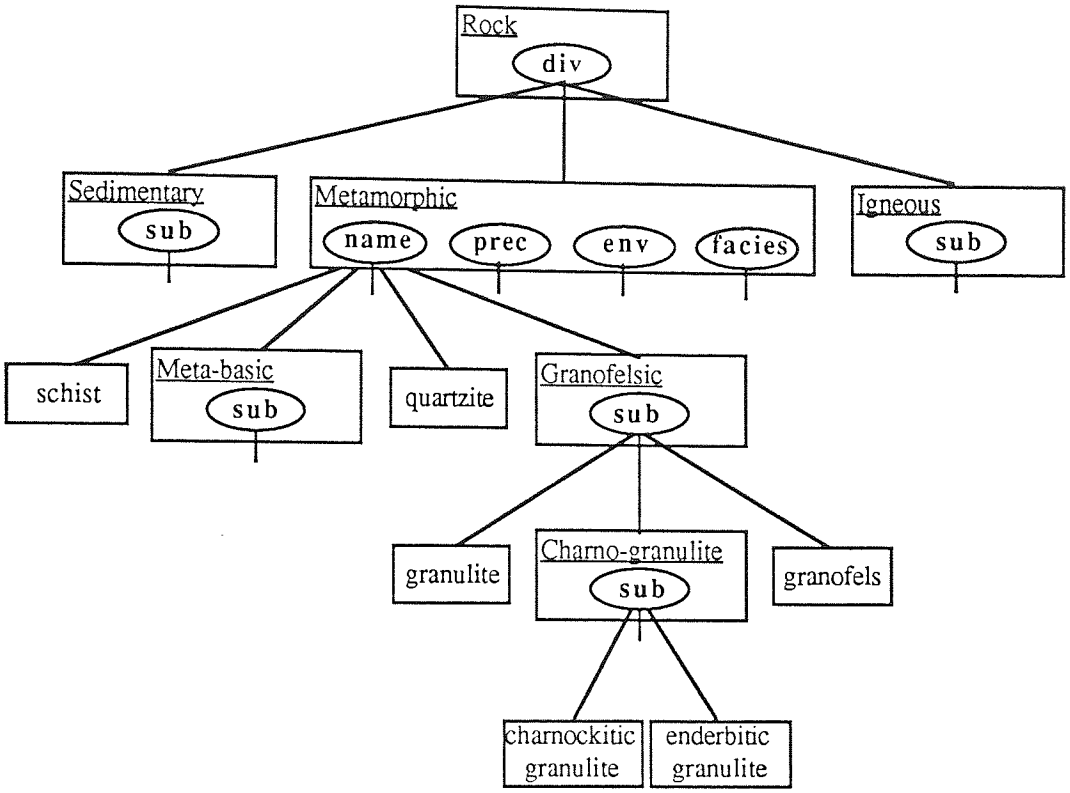

Figure 5.2 Example of the domain problem space in METEX showing high level control points and contained subclassification facets.

The METEX interpreter consists of five basic operational modules connected to a central control module as shown in Figure 5.3. This central module is used to interpret and execute the implicit and explicit control which has been built into a 'context frame' by the knowledge engineer and the domain expert. Control execution is carried out through the use of an agenda, local to the frame being interpreted, which contains a sequential list of the tasks which must be performed. These tasks are handed, from the front of the agenda, to the appropriate operational modules for further interpretation. Each module is in fact handed an identifier to the required knowledge base unit and is then able to recover this unit from the knowledge base. Essentially this communication between each module and the knowledge base is read/write as some of the information in the knowledge base units

may be altered during, and then reset after, a consultation. This alterable information should in fact be thought of as part of the dynamic database to which all modules are also given read/write access.

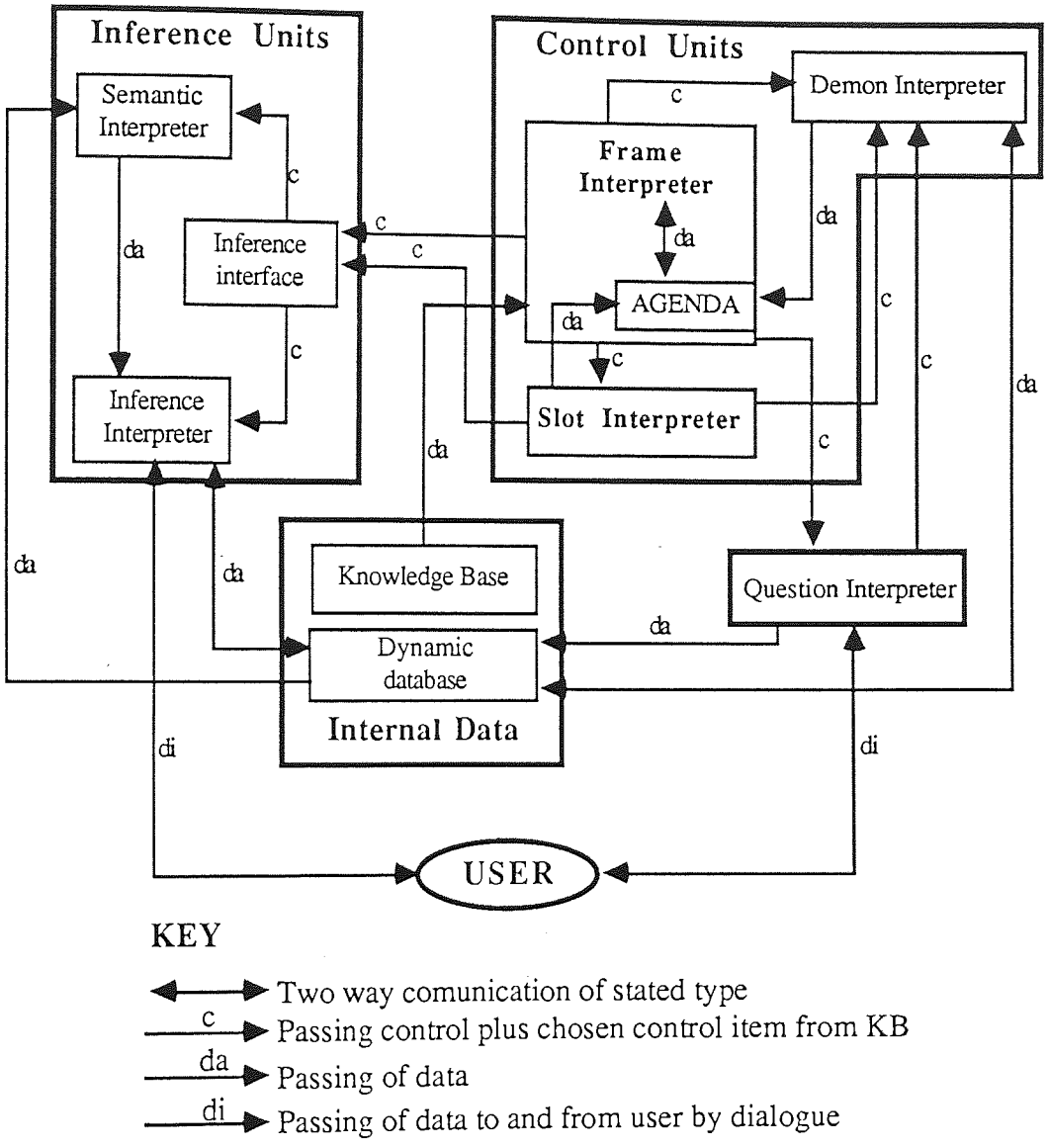

Figure 5.3 Schematic diagram of communication of data and control within the METEX interpreter.

Once the appropriate module has run, following the appropriate control information in the knowledge unit, control of the consultation is returned to the calling module which continues to work through its agenda. The agenda may or may not have been altered by the operational module to effect a change in the current strategy. Such changes to the agenda, if they occur, are not recognized as alterations by the interpreter since the control

module is only interested in reading the top item from the agenda, and passing control to the correct module. In this way the control module works through the tasks given to it until the agenda has been emptied at which point the system has completed interpretation of the current context.

5.2 3 Context Frames

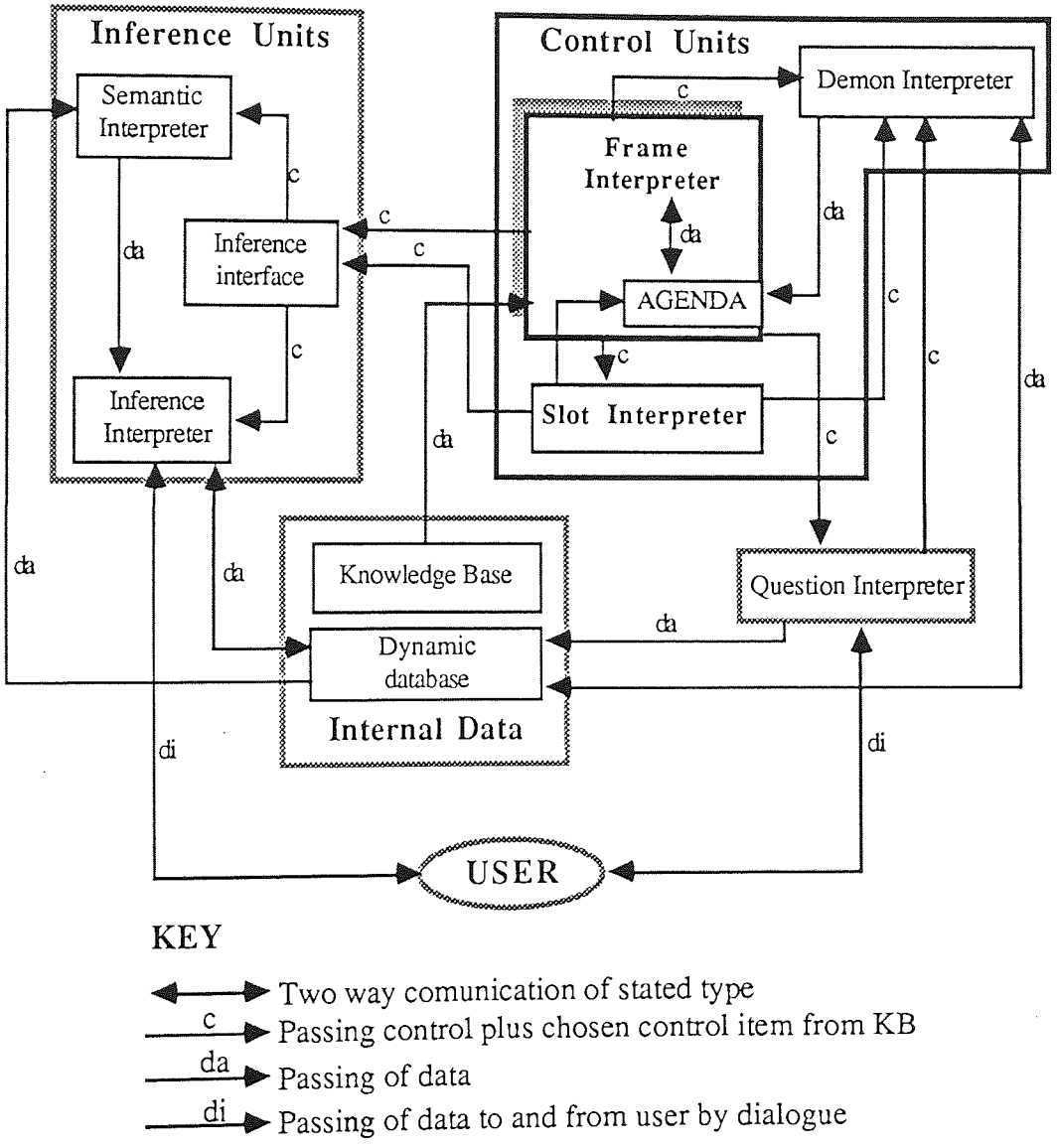

Figure 5.4 Communication of data and control diagram highlighting the frame interpretation module.

In METEX the main knowledge base control unit is the 'context frame'. Context frames are interpreted by the frame interpreter module, shown in Figure 5.4, which treats

each frame as a new level of control with its own control agenda. Context frames are designed to allow the knowledge engineer to organize the domain knowledge into conceptually distinct units. The context frame is used to encode control knowledge, or domain dependent meta-knowledge, separately from the inference knowledge, in a similar manner to that used by Aikins(1980, 1984) in her CENTAUR system. METEX context frames contain a list of slots which represent the tasks to be performed by default whenever the frame is activated. These tasks are performed in the order in which they appear since the frame interpreter assumes that the ordering represents implicit control information. This could be envisaged as forming explicit rules similar to those shown in Figure 5.5.

IF frame being processed
AND slot_1 has not been processed
THEN process slot_1
AND slot_1 has been processed

IF frame being processed
AND slot_1 has been processed
AND slot_2 has not been processed
THEN process slot_2
AND slot_2 has been processed

Figure 5.5 Example meta-rules capable of replacing control encoded within frames in METEX which use definition order as implicit scheduling of tasks.

As can be seen it is much simpler to allow for a certain amount of implicit control in this way than to explicitly encode the control knowledge. If the expert should however require to have conditions under which this implicit control is overridden, then this can be achieved by explicitly encoding rules to alter the agenda under such conditions. This facility is provided in METEX by the use of procedural attachments, or 'demons', which can be defined to cause a similar behaviour to that displayed by meta-rules. In the example frame layout in Figure 5.6 the demon ED1 is fired when the condition clause is evaluated to true. This results in the slot S3 being elevated to the head of the agenda. More complex

alterations to the agenda are of course possible if more complex procedures are entered in the action clause of the demons.

```
FRAME example_frame
 VIA  any_variable;
 SLOTS
  SLOT  S1
    ASSIGNS  a_variable
    FROM  rules_for_S1
   CAN_BE [a b c d e]
   ENDSLOT
  SLOT  S2
        :
   ENDSLOT
  SLOT  S3
        :
   ENDSLOT
 ENDSLOTS
 DEMONS
  D_  ED1 WHEN  [a_previous_variable  =  "value"]
        DO  [add_agenda("S3")];
  D_  ED2 WHEN  [slot_finished(S3)]
      DO  [syspr('slot  3  has  been  done')];
 ENDDEMONS
ENDFRAME
```

Figure 5.6 Example frame definition showing the combination of implicit and explicit control information. Implicit control is given by slot order, explicit control by procedural attachments, tested each time the system looks at the agenda.

It should be noted that the use of procedural attachment in various control units of METEX (see Section 5.2.3) allows the expert and knowledge engineer to provide complex procedures, which are specific to the domain, and to have these procedures run when the data and inferences dictate that they should. Returning to Figure 5.6, it can be seen that the demon ED2 will be fired when its conditional clause is true and will result in a POP11 procedure named 'syspr' being run to print its argument.

In each of these examples, the demon has evaluated to a boolean value and this is a requirement for all such rules. Since much of the data in METEX will have some uncertainty attached, it has been necessary to provide a set of predicate functions within the interpreter. These predicates operate by checking whether the certainty of their

argument, a data item, falls within the particular range of the predicate in use. The ranges used in defining these predicate functions, have been defined to have the same meanings as those used in MYCIN. Some useful adjective ranges have been described by Shortliffe (1976, 1984), and Harmon & King (1984, p.42) and can be drawn in a diagrammatic form as in Figure 5.7, where the adjectives used and their ranges define the predicates available within METEX. By applying these predicates to a data item in the METEX database it becomes possible to have a boolean result for the demons in order that they can fire. The reader should note that a further type of procedural attachment is provided within the inference module of METEX and is discussed in Section 4.2.1.

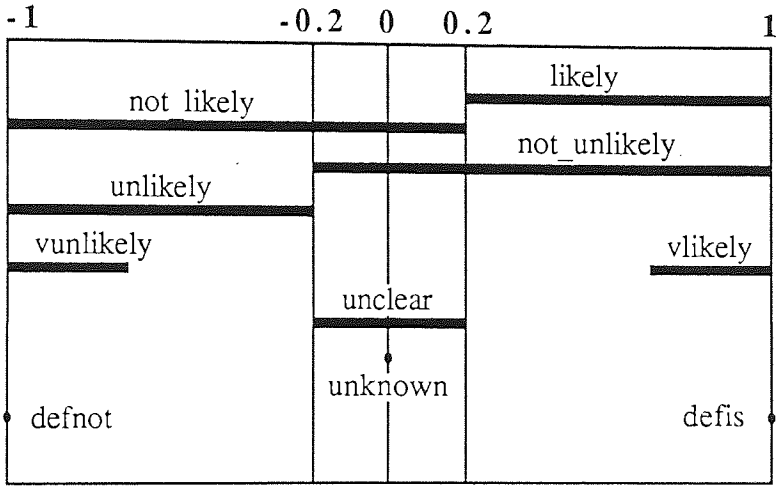

Figure 5.7 Predicates pre-defined for use in METEX.

Some frames in METEX have been used in a way which at first appears to be incorrect, one example being the 'granulite' frame in Figure 5.8. These frames do operate correctly and have been included for specific reasons. It is worth considering an example of the situations in which such a frame structure would be used as with the 'granulite' frame in Figure 5.8. This frame does not have any slots to be run as domain tasks. However, the frame, 'granulite' does have the effect of calling the frame 'charnockitic' since the procedural attachment demon, DFG1, will always evaluate to true as soon as the 'granulite' frame is activated. This results in the 'charnockite' frame being added to the agenda. After the 'charnockite' frame has been activated, and control has returned to the

'granulite' frame, the second demon, DFG2, is fired producing a suitable term for use in naming the rock.

```
FRAME granulite
 VIA  granofels_SC;
 DEMONS
   D_   DFG1 WHEN  [true]
            DO   [add_agenda("charnockites")];
   D_  DFG2 WHEN  [charnockite_SC  /=  false]
            DO   [make_result("gen_type",granulite_SC)];
 ENDDEMONS
ENDFRAME
```

Figure 5.8 Example 'granulite' frame used to activate shared subclassification rules by activating the 'charnockite' frame. This is achieved by firing DFG1 as soon as 'granulite' is activated.

Frames such as the 'granulite' example are, used as in this case, where two distinct contexts lead to the same frame. In the example the charnockite subclassification would be used if the rock were a gneiss or a granulite. In this case the charnockite result is used as a prefix to either 'gneiss' or 'granulite'. By using the frames in this way it is possible to avoid having to reproduce sets of rules which only differ slightly. However the technique is only applicable where the shared frame would be completely within context regardless of the path used to reach it. If the terminology within the shared frame is different depending upon the path used, that is depending upon the context in which it is used, then problems can and should be avoided by writing discrete frames.

5.2.2 Slots in METEX - Domain Tasks

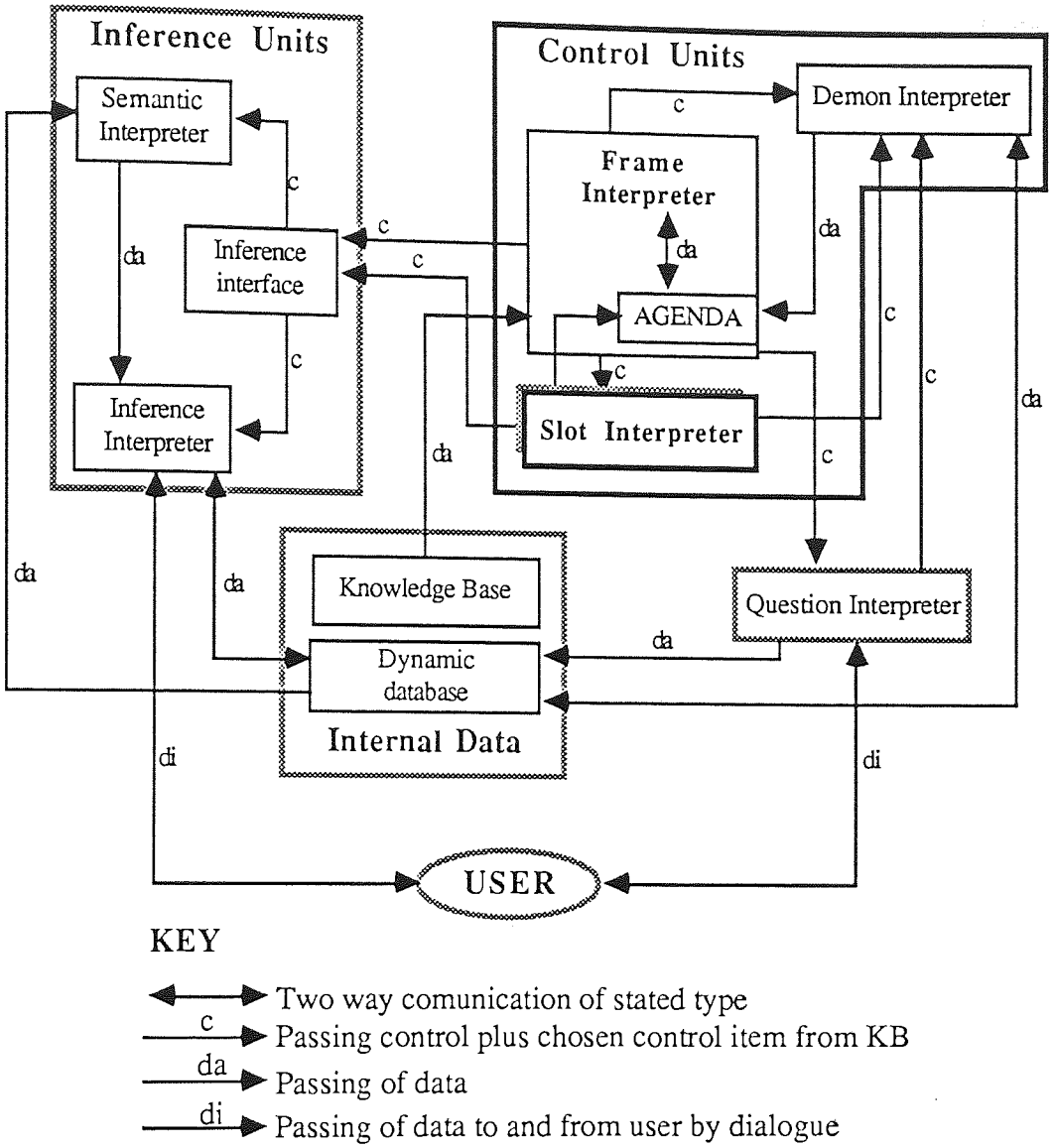

Figure 5.9 Communication of data and control diagram highlighting the slot interpretation module.

As described, METEX treats the slots belonging to individual frames as the tasks to be performed within the active context, each task slot being handled by the slot interpreter module (Figure 5.9). Every slot has a set of fields within it which are used by the interpreter to guide the control of the system. The priority control entry within a slot is the 'preactions' list as shown in Figure 5.10. This entry tells the interpreter which activities must have been carried out before the slot itself may be tackled. The interpreter checks each item in turn to ensure that it has been dealt with. Any actions which have not already been carried out are added to the agenda and control is returned to the main loop to process

these items. In returning control to the calling frame, the slot interpreter does not delete the active slot from the agenda. This ensures that the slot will be re-activated as soon as all of its preactions have been processed.

```
SLOT major_rock_division
 ASSIGNS  major_class
 FROM   major_division_rules
 CAN_BE  [igneous_rocks
          metamorphic_rocks
          sedimentary_rocks]
 PREACTS [Q1 Q2 Q3 Q4]
 DEFAULT  metamorphic_rocks
ENDSLOT
```

Figure 5.10 Root slot for rock classification context showing the use of preactions in evaluation.

Once all preactions are satisfied the slot itself is able to be considered. This is done by passing the list of goals, and the identity of the rule group to be used for inference, to the inference interface. Once the goals have been processed a set of results is returned and the value of the slot is assigned. Assuming that the slot is evaluated satisfactorily, then the system is able to continue on its current path of reasoning and the slot must now check its value against the list of frames in the knowledge base. If the name of any frame matches the value of the slot then this is taken as representing an implicit control instruction to evaluate the identified frame. The name of the frame, or slot result, is added to the front of the currently active control agenda. This ensures that the knowledge base is processed in a 'depth first' manner, with tasks being followed to completion before control switches to a new task. It should be noted here that if the expert prefers to use meta-rules to place a new frame onto the agenda, rather than having the names of his frames matching the correct propositions from the inference system, then he is free to do this.

In METEX the choice to follow a depth first strategy in the control level has been made in order that the user will be presented with apparently coherent lines of reasoning. However this gain must be offset against the increased inefficiency which is introduced to the search. This becomes important when, as in Figure 5.11, a context frame evaluates

three of its four tasks completely, only to discover that the fourth task fails forcing the system to backtrack. Had a breadth first strategy been used, as in Figure 5.12, then the system would have discovered the failure before having continued along the control paths below slots 1 to 3. However, this gain in efficiency results in the contexts being continually switched, as shown in Figure 5.13. Although this strategy might be efficient in terms of discovering failures, the user is likely to become very confused, since the continual switching of context will appear to be completely illogical.

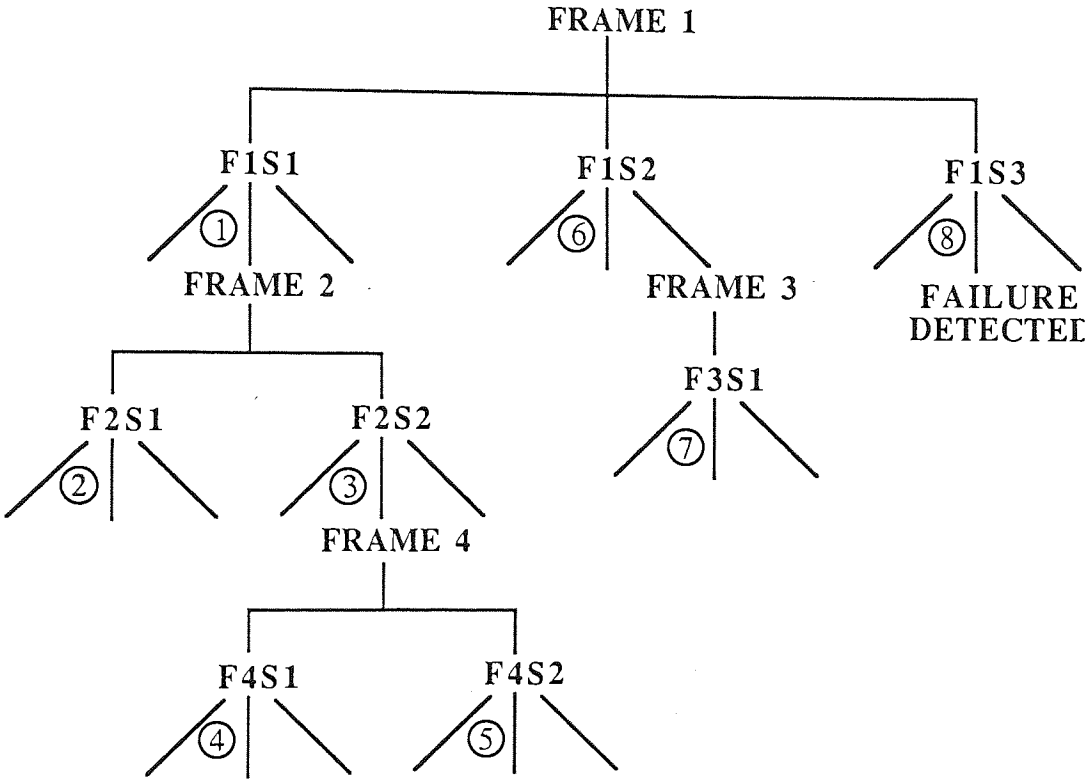

Figure 5.11 Depth first activation of slots showing wasted work if a failure occurs at a high level context within the search tree. Numbers indicate the order in which individual rulesets are activated. As shown depth first search requires eight rulesets before a failure is detected.

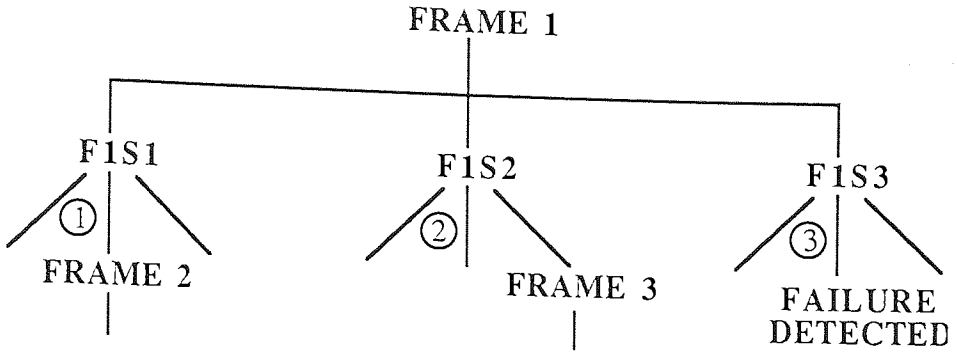

Figure 5.12 Breadth first activation of slots showing a reduction of wasted work before detection of a failure within a high level context. Only three ruleset activations occur before a failure is detected.

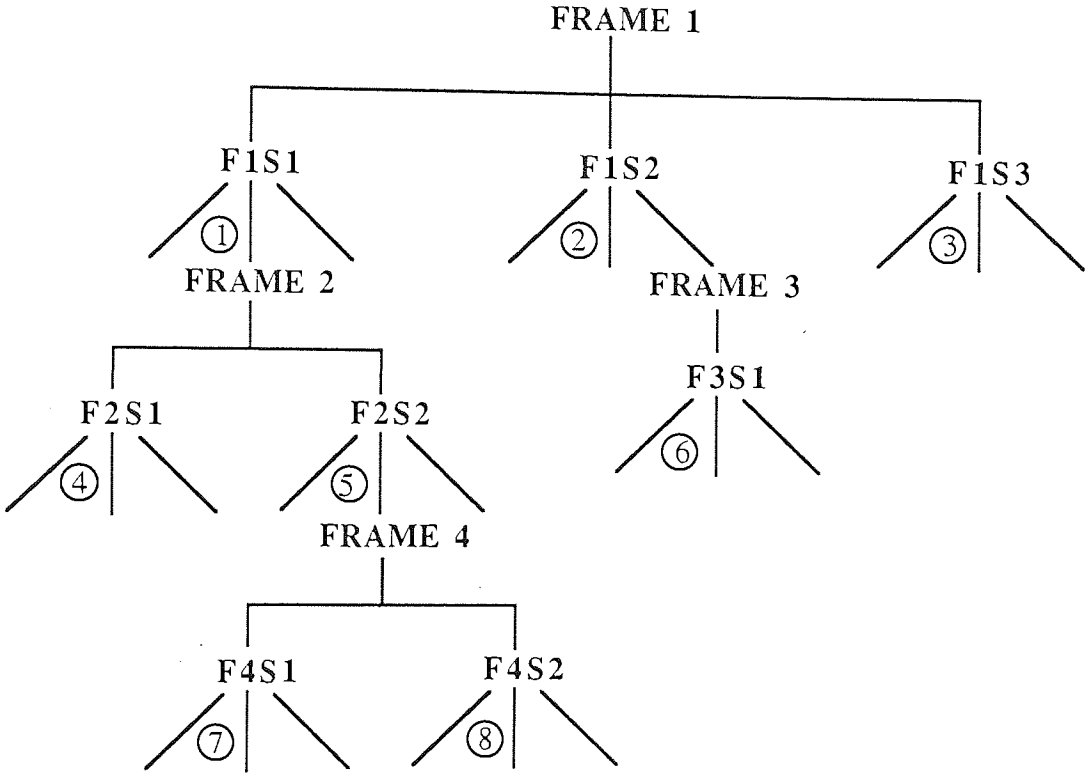

Figure 5.13 Example of apparent irrational behaviour in a breadth first search where no failure occurs. Although failures are detected more efficiently the system jumps around the knowledge base in a manner likely to confuse the user.

Occasionally it is desirable to concentrate on a line of reasoning while developing the knowledge base by assuming that everything falls into one of a set of contexts. In Figure 5.14 an example from METEX's knowledge base is shown where the topmost frame has been built as 'rock' which might lead to any of the sedimentary, metamorphic or igneous subdivisions. At this stage in the system's development it has proven desirable to have the

slot 'general rock division' cause the basic questioning about petrographic character to be asked. This data would be used in each of the major rock divisions and should therefore be gathered at this stage. However, the METEX system was originally intended for use in metamorphic petrology so that development has only been followed in that area. Rather than decide if the rock is metamorphic the system defaults to metamorphic by use of a default tag in the 'general rock division' slot. In this way the hierarchical position of the metamorphic rocks is maintained but the development of rules which are not required can be avoided.

```
FRAME rock
  VIA  top
  SLOTS
    SLOT major_rock_division
      ASSIGNS   major_class
      FROM    major_division_rules
      CAN_BE   [igneous_rocks
                metamorphic_rocks
                sedimentary_rocks]
     PREACTS  [Q1  Q2  Q3  Q4]
      DEFAULT   metamorphic_rocks
      ENDSLOT
ENDSLOTS
    :
```

Figure 5.14 Example drawn from the METEX knowledge base showing the use of a DEFAULT value in slot evaluation.

5.2.3 Question Interpreter

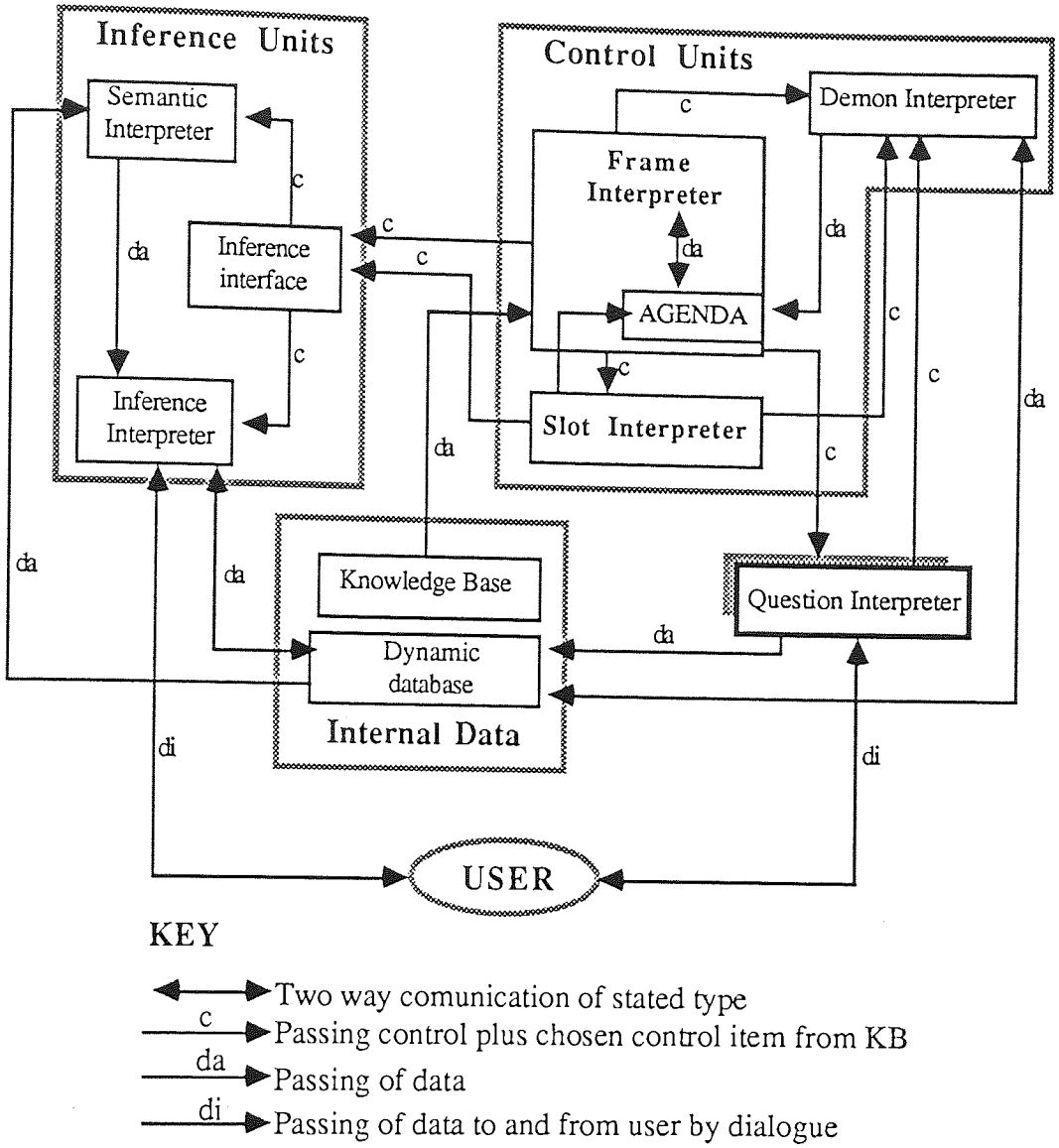

Figure 5.15 Communication of data and control diagram highlighting the question interpretation module.

When the agenda causes a question to be asked, control is passed to the question interpreter module, Figure 5.15, which interprets the question definition given by the expert. Questions fall into one of three main types in METEX. They are either 'simple' questions, to which the interpreter expects a yes or no answer, or 'menu' questions, to which a number corresponding to a menu item is expected, or finally a 'list' question to which the expert defines the expected responses. 'simple' and 'menu' question types are the least difficult for the system to interpret. Only one feature requires discussion here and that is the request for a certainty measure for any response given by the user. Both

questions are treated in the same way since the 'simple' question is a special case of a menu question choosing between yes and no. Once a response is given it is assumed to negate the other options, and the user is then asked how certain they are in their choice. The response here is limited to the sub-range [0 1], on the [-1 1] scale, since in making a choice the user has become committed to a positive belief in that choice.

The third question type in METEX is the 'list' question. This is provided since it is required specifically in the domain of petrology. In all of the petrological subdomains the geologist requires to know what the composition of the rock is. This is best represented as a list of minerals which are present, along with some estimate of the modal proportion of the mineral. In METEX questions of this type are handled by continually prompting the user to enter an item of data (in the example above a mineral name followed by a percentage), until the user enters a null item. As each item is entered the system checks it against the format entered by the expert or knowledge engineer. If the format is met then the item is accepted otherwise the user is asked to re-enter the data in the correct format.

In list type questions a feature of the POP11 matcher is used to allow the expert to attach some checking procedure to variables within the pattern matcher. This feature is used in the mineral list example from METEX to allow the system to check user abbreviations for mineral names against a list of accepted abbreviations so that the user is informed of any problems with his data.

As with the frame interpretation loop, the question interpreter is able to use demons to alter the agenda which is currently in control of the knowledge base interpretation. In this way it becomes possible for the questions which have been pre-defined to be asked in an intelligent manner, with one question potentially leading to another question, or indeed any other valid type of control item, being added to the agenda whenever the current data merits such an action. The choice of whether a new question is to be considered or not, would normally be made by a predicate test on the current data for the particular consultation.

5.2.4 Semantic Interpreter

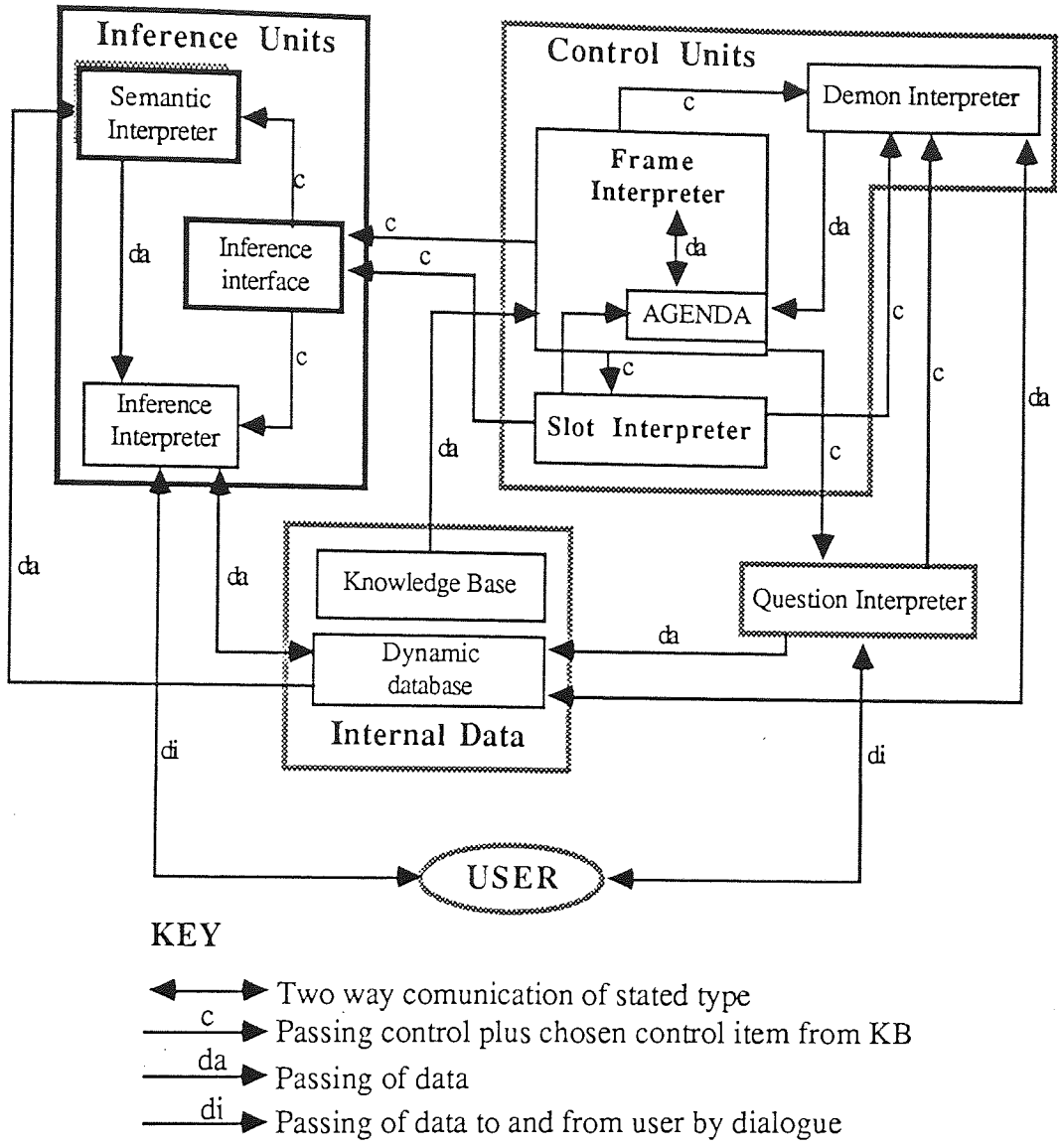

Figure 5.16 Communication of data and control diagram highlighting the semantic interpretation module.

The semantic interpreter built into METEX is capable of making simple associative links between words. Each node of the net corresponding to a meaningful domain word. All that the net itself does is to store the associations and to keep a record of the justifications for a word being 'in'. The justifications are set up when control is given to the semantic interpreter module, Figure 5.16. Essentially the semantic net nodes store three data items:

1. A list of the items to which this node leads hierarchically.

2. A list of the items which might possibly lead to the node.

3. A list of the items which have led to the node in the case in hand.

In order to make use of the semantic net the expert must provide a set of procedures which traverse the net in the way which is specific to his/her needs. For example Figure 5.17 shows a portion of the mineral semantic net in METEX. Let us assume that the system requires to know if the 'FeMg_silicate' accounts for greater than 40% of the rock. To do this the system requires to be able to find the amounts of the minerals which are linked to FeMg_silicate. These in turn have been inferred from other minerals which in this case come from the user. The system therefore recursively checks the daughters of the nodes required until it finds those which have come directly from the user. At this point the procedure is able to read the percentages of the minerals concerned and begin to return these to the calling procedure. Finally the system can check if the total for the node in question, gained by totaling the daughters, is greater than 40%.

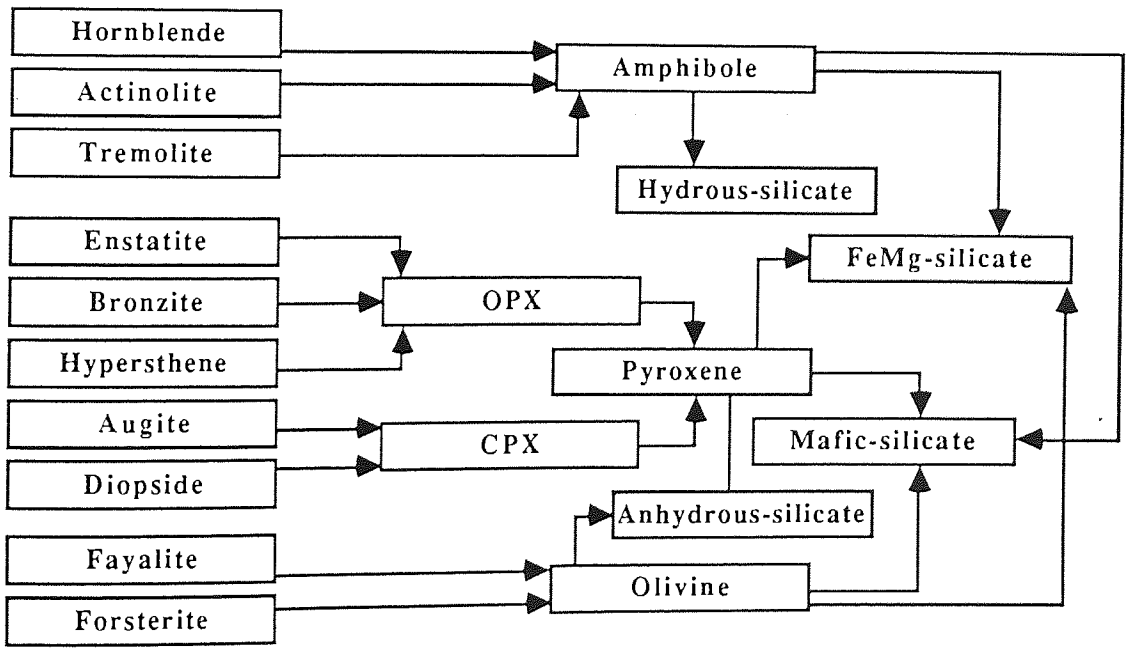

Figure 5.17 Sample portion of a semantic type net showing the relationships between mineral terms.

The procedures required for traversing and interrogating the net must currently be provided by the expert and knowledge engineer since they tend to be completely domain specific. The nature of the semantic nets in METEX makes it is possible to define multiple nets to store different conceptual groups of semantic information. For example it would be possible to store the mineral semantic net alongside a location net which gives the hierarchy of place names and regions around the world, along with the expected rocktypes, or metamorphic conditions, for that region. It should be clear to the reader that the structure available in METEX for semantic information is based on a simple IS_A with the knowledge engineer being expected to provide the necessary traversal routines for the domain.

5.2.5 Inference Interface

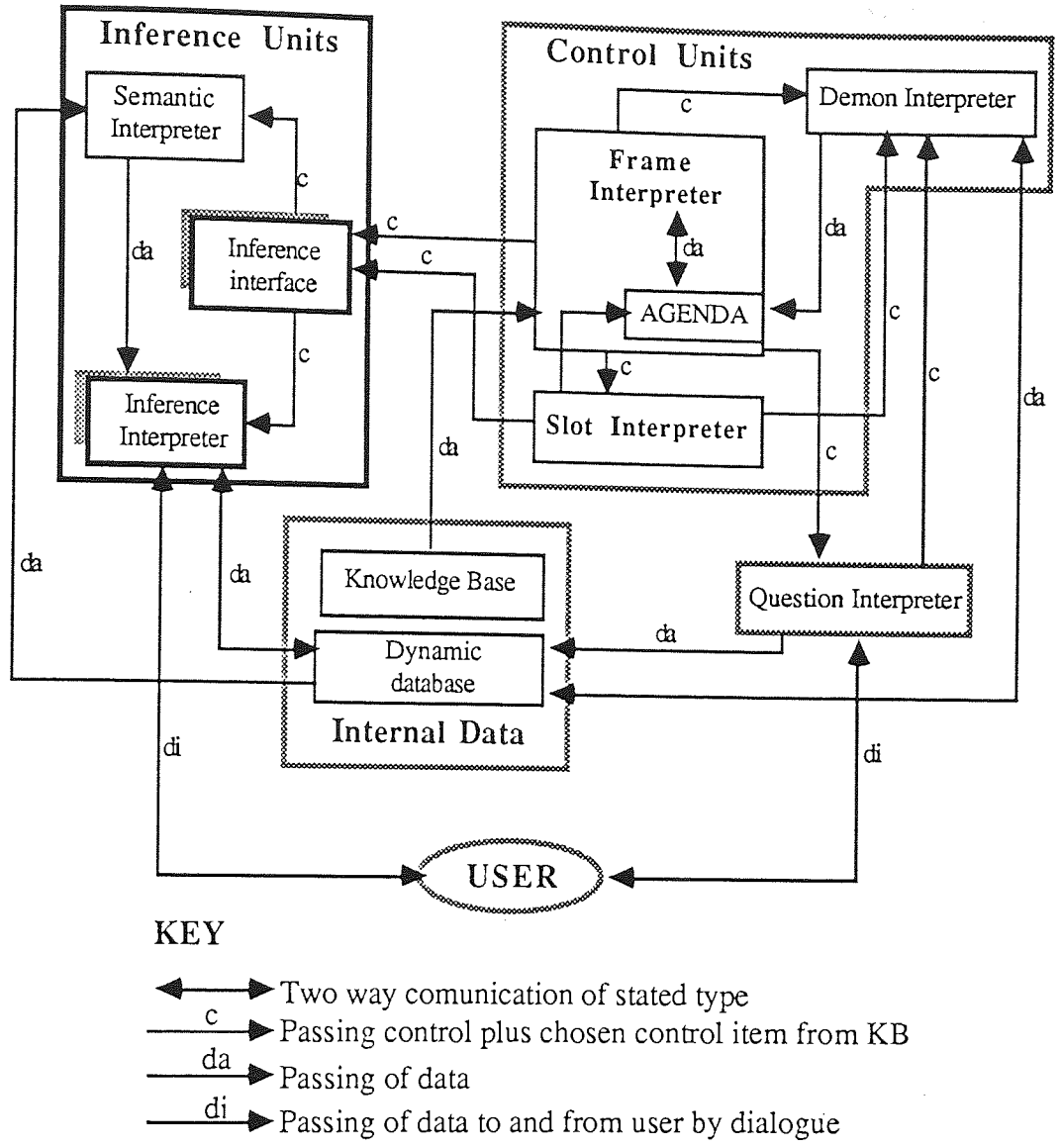

Figure 5.18 Communication of data and control diagram highlighting the inference interface and rule interpretation module.

As described above (Section 5.2.2), each task slot carries information giving directions as to what set of rules to use for its inference and what the possible goals are. These two pieces of information are passed to the inference interface routine, Figure 5.18, which activates the required ruleset giving the inference interpreter, in this case Augmented Eshell (see Chapter 4), the list of required goals. However, before control is given to the inference interpreter the interface must first check all of the propositional clauses used in the ruleset against the dynamic database. In this way any propositions which might have

been inferred in a previous ruleset, run during the current consultation, can be automatically initialised in the new ruleset.

Once control has passed to the inference interpreter, it will run to completion as described in Chapter 4. The inference interface is then given a set of certainty factors attached to all of the propositions in the activated ruleset. The first task is then to assert these propositions in the dynamic database for future reference. Finally the set of possible goals is analysed to discover both the result of the inference, and also some measure of the reliability of that result with reference to possible backtracking requirements in the future, (see Section 5.3). The inference results are then able to be returned to the slot interpretation routines for processing as described in Section 5.2.2.

5.3 Backtracking Procedure

Backtracking capability is an important feature of any search based system. Without backtracking it would be impossible to correct mistakes made in choosing paths towards the final solution. For this reason a method for allowing the system to decide both when and how it should backtrack has been introduced to METEX.

In order to decide that backtracking is necessary a system must be able to reason that the path or branch which it has chosen has in fact failed. The system is then able to backtrack to the point at which the failed branch was chosen, and choose an alternative branch from that point. In METEX the major difficulty is in deciding when, in fact, a chosen branch has failed. If we consider Figure 5.19, it is clear that in METEX the use of multiple tasks leads to a search space which is in fact a set of overlain search trees. In order to recognize a need to backtrack in such a search space METEX must first consider the large scale tree structure. In this the important features are the frames themselves. METEX must decide whether a frame which has been chosen is a success or a failure in terms of its evaluation. This decision must in turn be based on the success or failure of the individual small scale tree units which develop under each task slot. A task slot must be

deemed to have failed whenever it is unable to find a possible branch to follow in the tree developed below the task. This occurs when, for instance, a ruleset called by the slot finds all of the possible results to be false, with partial failure being shown by some real degree of disbelief (i.e. CF values between 0 and -1), in all of the possibilities.

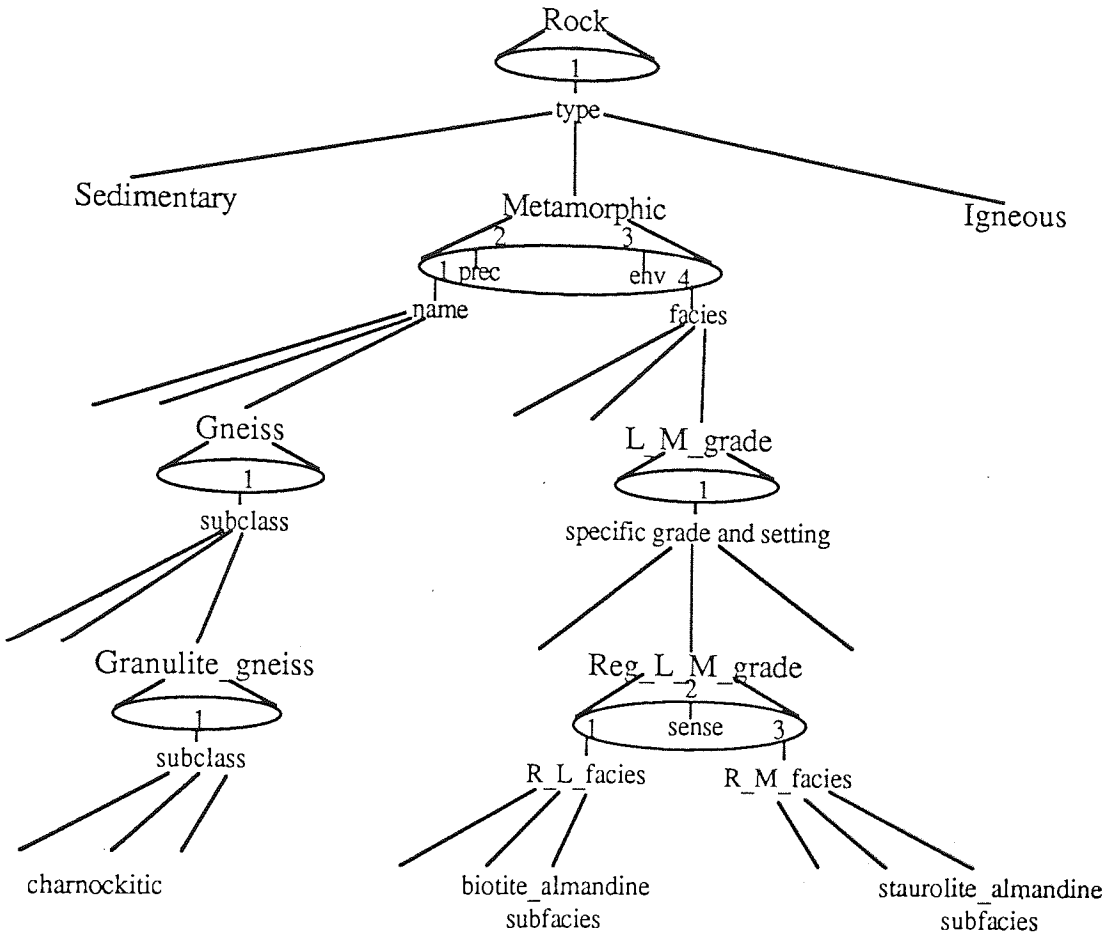

Figure 5.19 Schematic showing the overall structure used in the search space of the METEX knowledge base. Ellipses indicate the slot tasks within a frame with order of activation indicated. Consistency with the knowledge representation is provided by using similar ellipses where only one task occurs.

When a slot has had its possible outcomes evaluated by the inference interpreter, the best result is chosen as the value for the slot. The upper bound on the certainty range for that result is taken as being the measure of the success of the slot itself, thus linking the success of the slot directly to the system's success in choosing a possible branch path.

Once the system is able to maintain some measure of the success or failure of individual slots within a frame, it becomes possible to gauge the relative success or failure of the overall context frame. It is important to somehow monitor the performance of the frame continually since if one slot fails completely it becomes doubtful whether the frame should have its evaluation continued through the remaining slots. At this point it becomes very unclear what the strategy of the system should be. For example, suppose that METEX were in fact capable of handling rocks from all three major rock divisions, igneous, sedimentary and metamorphic, rather than only the metamorphic group. If such a system were confronted with a coarse grained, crystalline, predominantly calcium carbonate rock with some impurity (a rock description which could fit any of the divisions based on this information alone) what should happen? Let us assume that for some reason the system marginally prefers to place this rock in the metamorphic field. The rock will then be classified as a marble with 100% certainty within the context of the metamorphic rocks. The system would then go on to assume that a limestone precursor is likely thus providing two completely successful tasks in the metamorphic context frame. If the system were now to attempt to place the rock in a suitable metamorphic environment, but found that all of the possible metamorphic environments were false, what should be done? In this case the facies slot will not have been evaluated so could still possibly succeed or alternatively it might fail.

In METEX such a situation is dealt with by allowing the system to assume that when unevaluated, any slot has a possible certainty of 1 (Figure 5.20a). Now as the slots are evaluated METEX combines their resultant certainty factor with those of all of the other slots. This gives a resultant value which indicates the maximum possible combined certainty for the whole frame. Since a MYCIN type certainty system is used throughout METEX the combination is done using the three MYCIN combination functions described in Section 2.2.4.2 and Section 4.2.3. An arbitrary 0.2 threshold is used at this point to mark the assumed failure of a frame, the 0.2 value being chosen purely on the basis of its use in MYCIN. Using this threshold, the metamorphic context frame in the example above

will be considered to have failed completely since the combined certainty of the frame will come out as a 0 (Figure 5.20b). It should be noted here that the MYCIN combination functions strongly favour CF values of 1 and -1 as these can only be reduced by a directly conflicting -1 or 1 respectively, so unless an early slot in a frame fails completely, or else all of the slots have been evaluated with CFs not equal to 1 and combining to a value less than 0.2 (Figure 5.20c) will a frame be considered to have failed causing backtracking to begin.

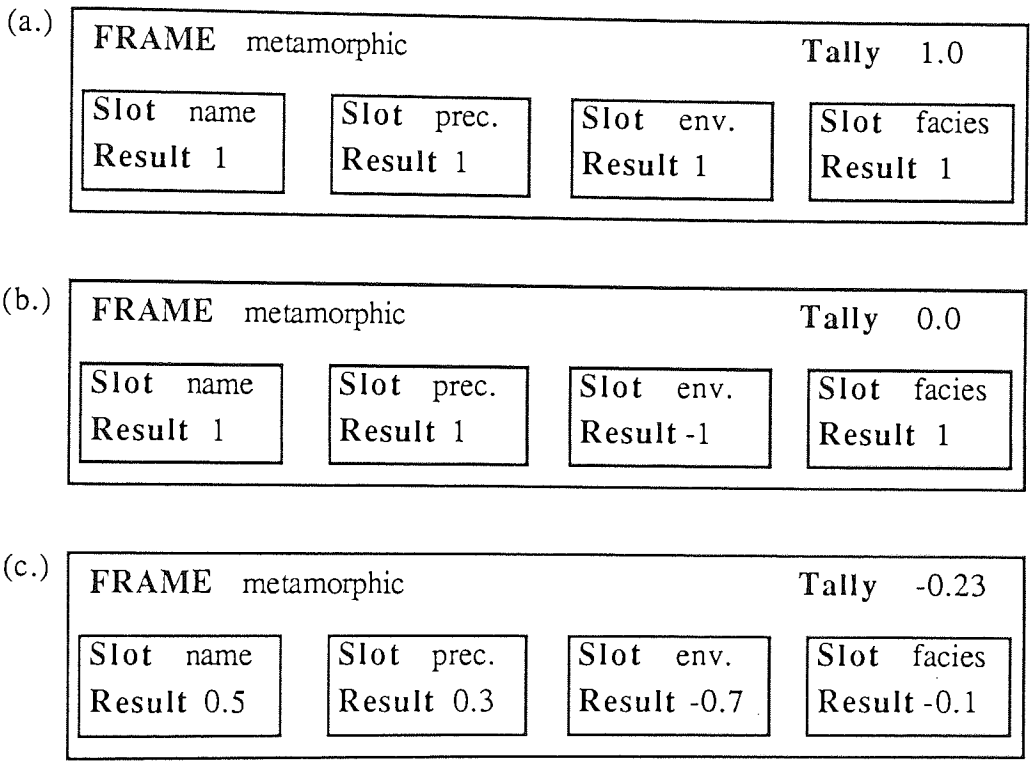

Figure 5.20 Example showing evaluation of a frame tally to decide when the system ought to initiate backtracking. a) shows the tally either before any evaluation (when slots have a default of 1.0) or after complete evaluation of all slots to 1.0. b) shows failure at the environment slot causing failure of the frame avoiding wasted time on the facies slot. c) shows all slots requiring to be evaluated before the frame is decided to have failed.

Once a frame has decided that it has failed it becomes necessary to discover whether there is a suitable target, in the search space, to which control might backtrack. METEX makes this decision by progressively moving back through the search path considering the "solidity" of the chosen frame. If a frame was chosen by a slot which had no possible

competing results then there is no point in backtracking to that slot. In this case the system must then move back and examine the frame to which that slot belongs. However, when a slot evaluates to have more than one possible solution, that is more than one of the solutions has been shown to have a positive CF, METEX uses the best result to decide which of the possible branches to follow. Obviously this slot becomes a potential target for backtracking since other branches, which could be followed, do exist (Figure 5.21). Once this slot has been found METEX backtracks to it and chooses the next best result from the inference system. The certainty of this result is then used in combination with the certainties of the other task slots in the frame to decide whether or not backtracking must continue to an even higher level in the hierarchy. If the combined value of the slots is greater than the 0.2 threshold then the new branch will be attempted as a solution path.

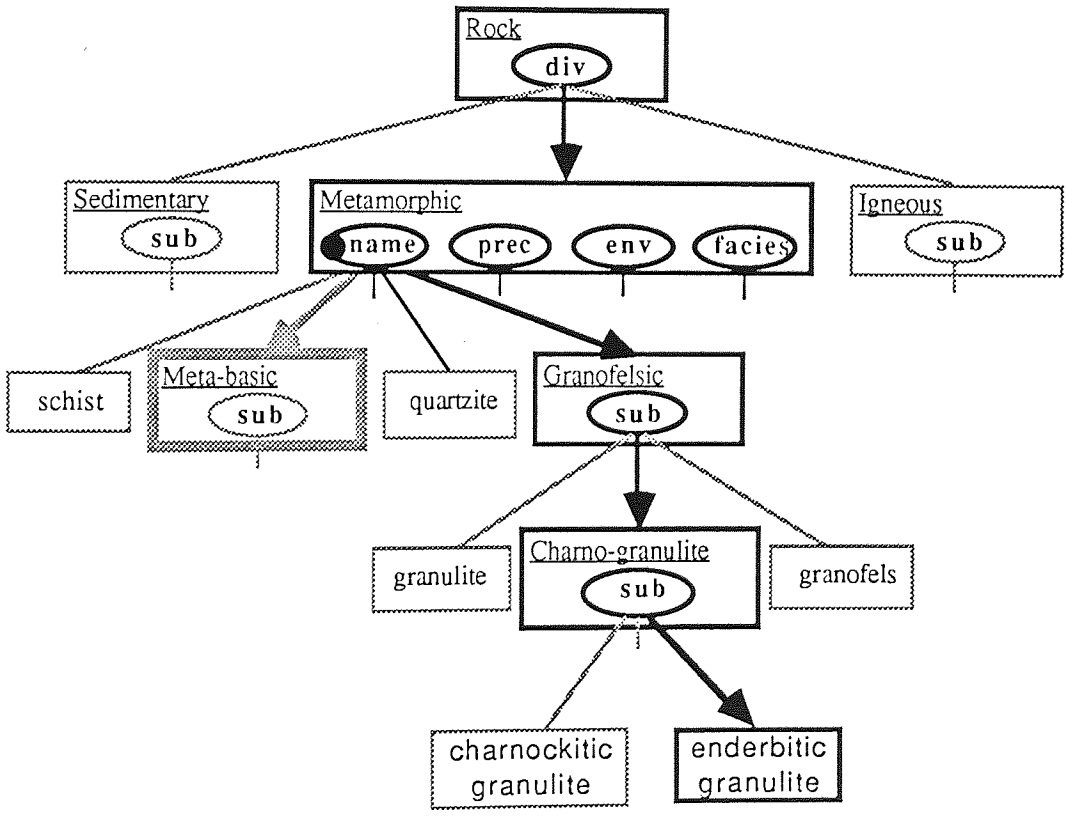

Figure 5.21 Example of successful attempt to find an alternative to a failed branch through backtracking.Heavy lines show successful path, thick dotted lines show failed path, light dotting shows unavailable branches. The backtrack target is flagged by a black dot here.

When a context frame decides that it has failed it searches for a possible point in the search space to which control can be passed by backtracking. A new problem arises when no possible targets exist in the solution path chosen so far. In such a case all of the frames down from the current one have been chosen by task slots which have evaluated with only one likely result (Figure 5.22). This means that as far as its reasoning is concerned, METEX has made no mistakes in its previous choices. The system must therefore assume, in a manner of speaking, that its knowledge, in the context of the frame which has failed, is incomplete. Such an assumption may of course be untrue. Firstly it is possible that the knowledge used in previous frames was incorrect, and secondly it is most likely that it is only the knowledge in the failed slot which is incorrect. It is debatable quite what should now be done by the system. METEX will currently simply warn the user that it is unable to resolve the task slot in question although the context itself is believed to be correct.

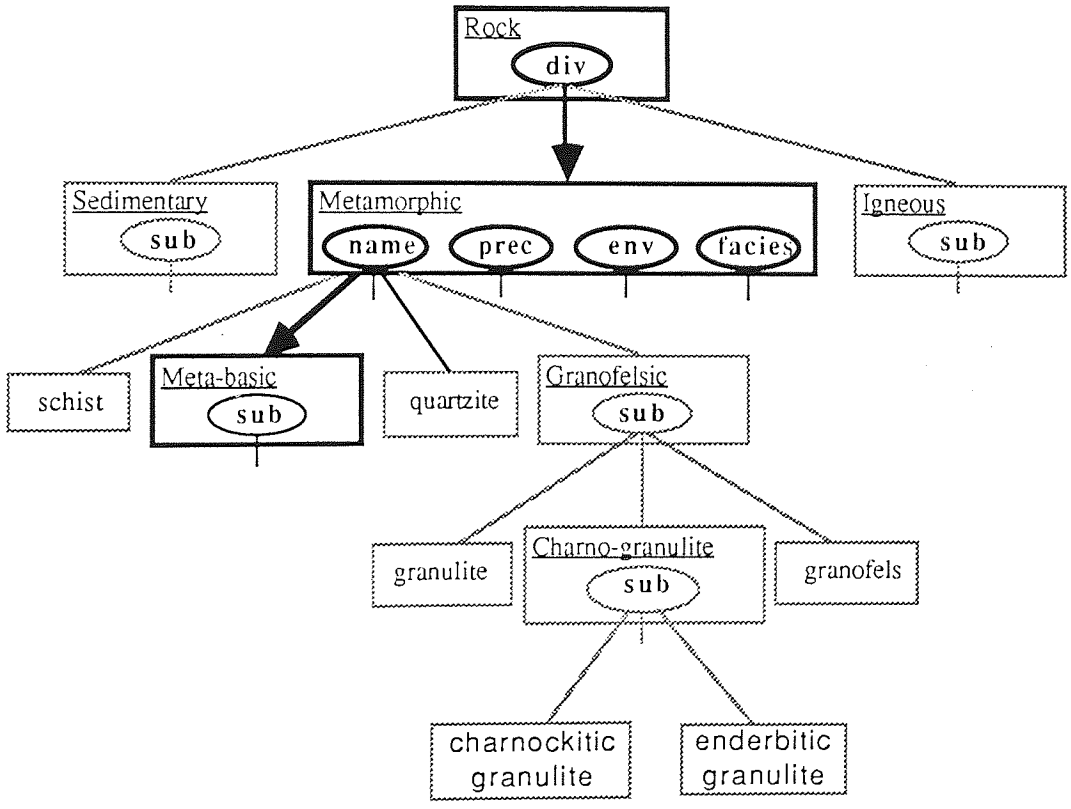

Figure 5.22 Example of attempt to backtrack where no alternative branch exists. System will exit with meta-basic as best result possible in the name task.

Returning to the carbonate rock example given earlier, METEX will attempt to backtrack to the choice of major rock division. If this is possible, that is the choice of metamorphic context was clearly dubious, then evaluation will be continued in whichever rock division was next most likely. However, if the rock could only be metamorphic then the system will simply report that it is unable to decide on the metamorphic environment. The system will then attempt to classify the metamorphic facies before ending the consultation. Results will be summarized in such a case by indicating that the rock is a marble, that it was once a limestone, that the environment was unresolved along with whatever decision was reached regarding metamorphic facies.

CHAPTER 6

The Metamorphic Petrography Knowledge Base

6.1 Knowledge Base - Design and Purpose

The knowledge base of METEX has been designed to serve a dual purpose. Firstly it has to be capable of providing the information required to collect data about a particular rock specimen and, through that data, arrive at a classification of the specimen. Secondly, the knowledge base should contain enough additional information to ensure that the novice user will leave a consultation with some general information about the rock type in question and perhaps some further reading. It is also to be hoped that if the knowledge base is designed to gather data about basic, important rock features at the correct times, then the habit might transfer to the novice. Such a system might be considered as an automated check-list. This is in fact more important than it at first seems. An early test of novice, in this case undergraduate geologists, showed that, when left to freely describe a specimen, there was a marked tendency to omit data which would be recorded as matter of habit by the more experienced geologist. However, when suitable prompts were given, the data about features which were missed earlier was gathered without difficulty. It seems therefore to make sense that every time a novice uses a system such as METEX the procedure of always looking for certain features should become a habit.

The knowledge base of METEX has been designed to represent all of the domain specific information within the overall expert system. In this way the distinction between domain knowledge and the more generally applicable interpreter routines is clearly defined and maintained (Figure 6.1). The system has been restricted to handle only the metamorphic rocks as this is a broad and complex enough division of rocks to allow a reasonable test of the approach chosen. The system in fact starts at the contextual level of rocks in general, but assumes that any specimen must be metamorphic. This step has been taken in order to provide the branching points for sedimentary and igneous rocks for future development. In order to allow the system to handle these divisions it would be

necessary to provide a context frame to control further classification within each of the contexts each being activated through use of a suitably defined set of classification rules, a task which in itself is non-trivial. Some features of the knowledge base would in fact be common to all three major rock divisions. These features are provided here and should be of assistance in future development of the system to cover the overall petrographic domain.

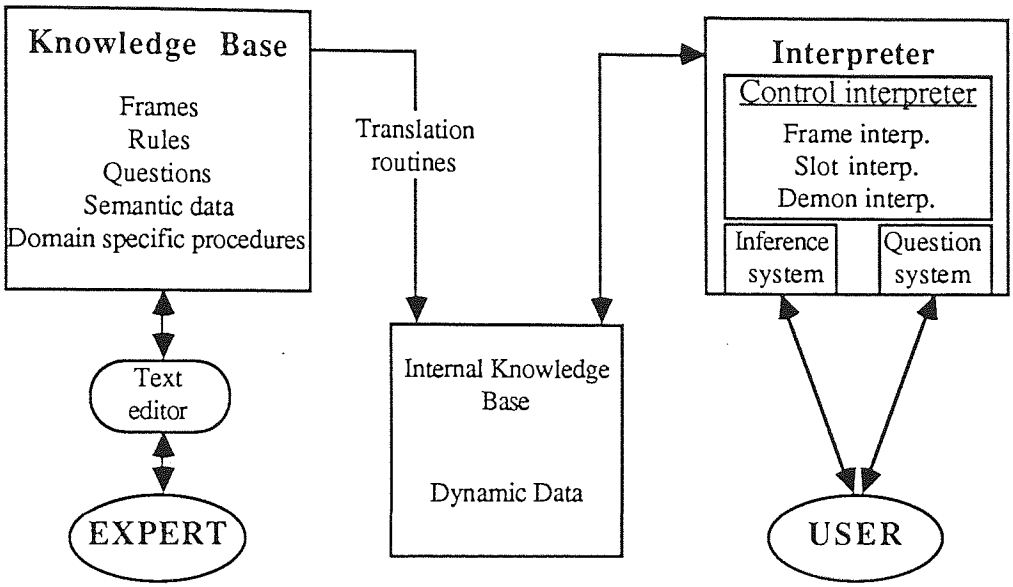

Figure 6.1 Overall architecture of the METEX system showing distinction between knowledge and interpreter.

The design of METEX makes certain assumptions about the end user. The main assumption is that the user possesses some basic knowledge of geology in order that basic textural features and modal compositions can be obtained. The most important point here is the need for the user to identify the minerals in the rock. If these are mis-identified then any classification given by the system will possibly be off mark to a degree determined by the importance of the mineral which has been mis-identified. Unfortunately no solution is available to this as it would require a further expert system specific to identifying minerals, a problem which does not fall within the scope of this project. However, if a full petrographic assistant system is ever to be realized then it will be necessary to develop a knowledge base capable of identifying the basic information such as textures and mineral species as well as the more important rock types. Such a system might have an architecture

as in Figure 6.2. A mineral identification system has already been developed by West (1985) so a suitable knowledge base for this task already exists. However, extensive modification would be required in order to interface this with METEX since the system uses a different knowledge representation and is written in Prolog.

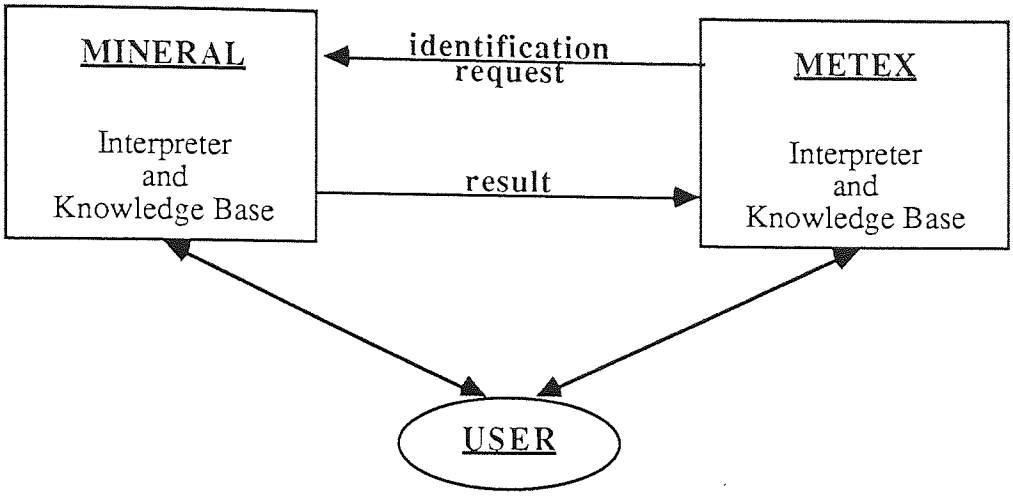

Figure 6.2 Possible high level architecture of a system capable of identifying all rock types and assisting in mineral identification.

6.2 Knowledge Base Features of General Petrographic Use

As stated above, some features of general use in the petrographic domain have been incorporated into the knowledge base of METEX. Although these features have been incorporated they have only been developed as required for metamorphic rocks and will require to be augmented to allow for use in igneous and sedimentary rocks. The highest level context frame in METEX is the general 'rock' frame. This frame would control classification of any rock type in a full petrographic system. The frame has been included in this more restricted metamorphic system simply because some features, particularly textural and compositional ones, are important in classifying all rocks. In order to gather data about such features at the correct time it is natural to include a suitable level of abstraction to allow this and to that end the 'rock' frame, which forms the root to the whole petrographic classification tree, has been partially built.

A further feature common to all of the major rock divisions is the use of semantic type associations relating minerals to their hierarchical groups, both in terms of nomenclature and chemistry. As regards METEX this is particularly important in terms of the system appearing intelligent. It would almost certainly deter any user if, whenever they gave a mineral, say augite, in the modal composition, the system later asked if there was a pyroxene present, or if an iron magnesian silicate were present. In order to represent this information, a list of the major rock forming silicates has been provided along with the major associations generally made between these and their hierarchical groupings. The knowledge base also contains some domain specific procedures to allow the use of standard abbreviations, which are converted to their correct terms for use in the above association net, as well as suitable functions to arrive at totals for particular groupings. For example, the expert might wish to write a rule which considers whether the rock is predominantly FeMg silicates. To handle this the system first finds how much FeMg silicate is in the rock, by checking systematically through the instances of FeMg silicate in the net in order to find a cumulative total. The net has been kept as simple as possible with hierarchical groupings being carried out only where they have a geological meaning taking care to ensure that any basic mineral node has only one path to any of its superior groupings, otherwise it would be possible to add the amount of a mineral to a cumulative total more than once.

Having achieved a total, the system then checks this against the defined numeric meaning of the term 'predominant', represented as a fuzzy set distribution. This is an important feature of METEX, since the expert is required to give some indication of the fuzzy set distribution across a term of this type. For example, with the term 'predominant' we might say that the mineral, or mineral grouping, should account for greater than 80% of the mode. However, if we consider a rock of 77% quartz this could still be classed as a quartzite, even though quartz is not predominant according to the 80% definition. In order to allow for this it is necessary to attach certainty factors, as described in Section 2.2.4.2, to small ranges of numbers within the definition of a term such as 'predominant'.

Whenever a number falls in a range defined in the term, then the associated CF is returned for the test. Consider if 'predominant' were defined as [82 100] = CF 1, [77 81] CF 0.5 [74 76] CF 0.2 [0 73] CF -1. If we gained a total of 77% for the mineral or mineral group of interest then a value of 0.5 would be returned for the test. This is not a mathematically accurate translation of the fuzzy set, as defined by Zadeh (1978), which should in fact have a smoothing function attached in order to find the CF as demonstrated in Figure 6.3. However, the technique used here does serve as a relatively simple method for relaxing the boundary conditions on fuzzy or subjective categories.

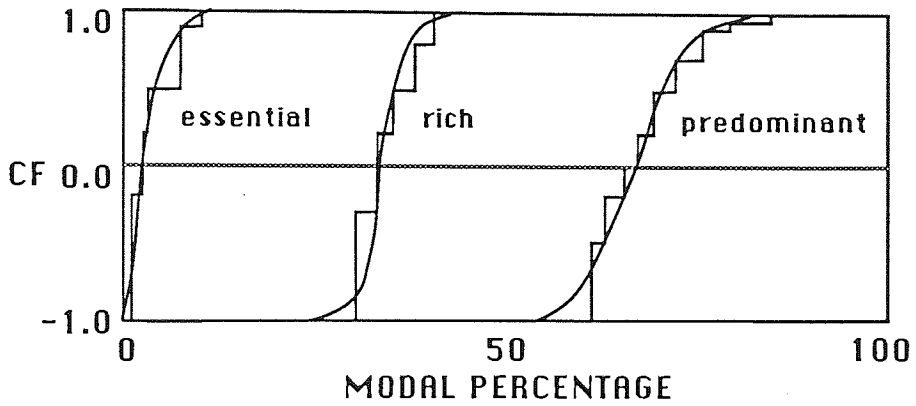

Figure 6.3 Comparison between ideal smooth fuzzy set functions and the approximation as used in the METEX knowledge base.

6.3 Metamorphic Rocks Context Frame

The knowledge base of METEX, as described above, is concerned with the context of the metamorphic rocks. This context is controlled by the context frame 'metamorphic_rocks' which attempts to model a methodical approach to the problem of classification. The expert tackles metamorphic classification as a series of different, but related problems. When watched closely the expert appears to tackle these problems in a different order for each specimen switching between problems with ease. In METEX these sub-problems, apparently tackled by the expert in parallel, are tackled in sequence. The

sequence has been chosen based on the need for information from one task in the satisfaction of another. The order chosen for the tasks is as follows:

1. GENERAL NAME:- The classification of the rock to a descriptive name which can transfer both compositional and textural data to another geologist. For example: schist, gneiss, hornfels, amphibolite or marble.

2. PRECURSOR NAME:- The classification of the rock into a general type which conveys the nature of the pre-metamorphic rock. For example: pelitic, calcareous (limestone or marl), sandstone.

3. METAMORPHIC ENVIRONMENT:- The environment in which the metamorphism of the rock occurred. This is currently restricted to regional and contact, but should also cover metasomatic, cataclastic, burial and sea floor metamorphism.

4. FACIES CLASS:- The classification into a descriptive term for the conditions of pressure and temperature at which metamorphism occurred. This task relies largely on information found in each of the others so is best handled at the end of a session. A facies term can only convey the general field of P and T, within which a rock must have been metamorphosed, rather than exact values.

Each of these tasks will be considered in turn in this chapter, discussing the different available solutions within each task, paying particular attention to those rock types which caused difficulties.

6.3.1 General Name

The general name of a metamorphic rock, as used in METEX, is the name which would be used by a geologist to convey some indication of its petrographic character. As previously described (see Section 2.1.1.1) arriving at the correct solution within this task

can prove difficult. The major problem arises because the terms used in the classification are dependent upon features of either composition or fabric. This allows a certain amount of overlap to exist which the geologist resolves with intuitive judgement. The general guidelines set out by the SubCommission for Systematics in Metamorphic Rocks (SCMR) in 1985 (Athens) noted that the difference in criteria used in each of the broad groups of rock names allowed such an overlap, and suggested that this should not be a cause for concern. It was also suggested that broadly descriptive terms such as 'schist' or 'gneiss' should not be restricted by modal or structural criteria, allowing them to cover a wide range of the many possible combinations of features. To make the terms more specific however, the use of prefixed mineral names and compositional or structural modifiers is encouraged. In METEX the classification of general names is carried out by assuming that a compositionally based name is to be used whenever correct. This name can then be prefixed whenever it is necessary to convey additional textural information. This choice of strategy has been made since compositionally bound terms tend to be more specific than fabric based terms. A fabric based name can then be used whenever it is the only option and suitably prefixed by important constituent minerals or specific compositional descriptors, for example, 'garnet biotite schist' or 'charnockitic gneiss'. The only major problem then arising is the choice of prefixing minerals. The SCMR guidelines state that the minerals used for qualifying a name should not be essential in the definition of the general name chosen. Mineral names should be added where they account for greater than 5% of the mode, in increasing modal importance. Minerals of less than 5% should be added as for example 'biotite-bearing'. In METEX the minerals are added in a manner only slightly varied from this. Minerals of less than 5% are put in first with the qualifying term '-bearing' after them. This is then followed by the minerals of greater than 5% occurrence, plus those of particular importance such as 'kyanite', regardless of modal percentage. Currently this prefixing is carried out by adding all minerals not of essential importance in the definition of the main rock name. This is as recommended by SCMR; although there is something to be said for restricting the number to four, a suggestion

recognized by SCMR. This system allows METEX to come up with an almost infinite variety of names to cover the wide variation possible in the features of metamorphic rocks.

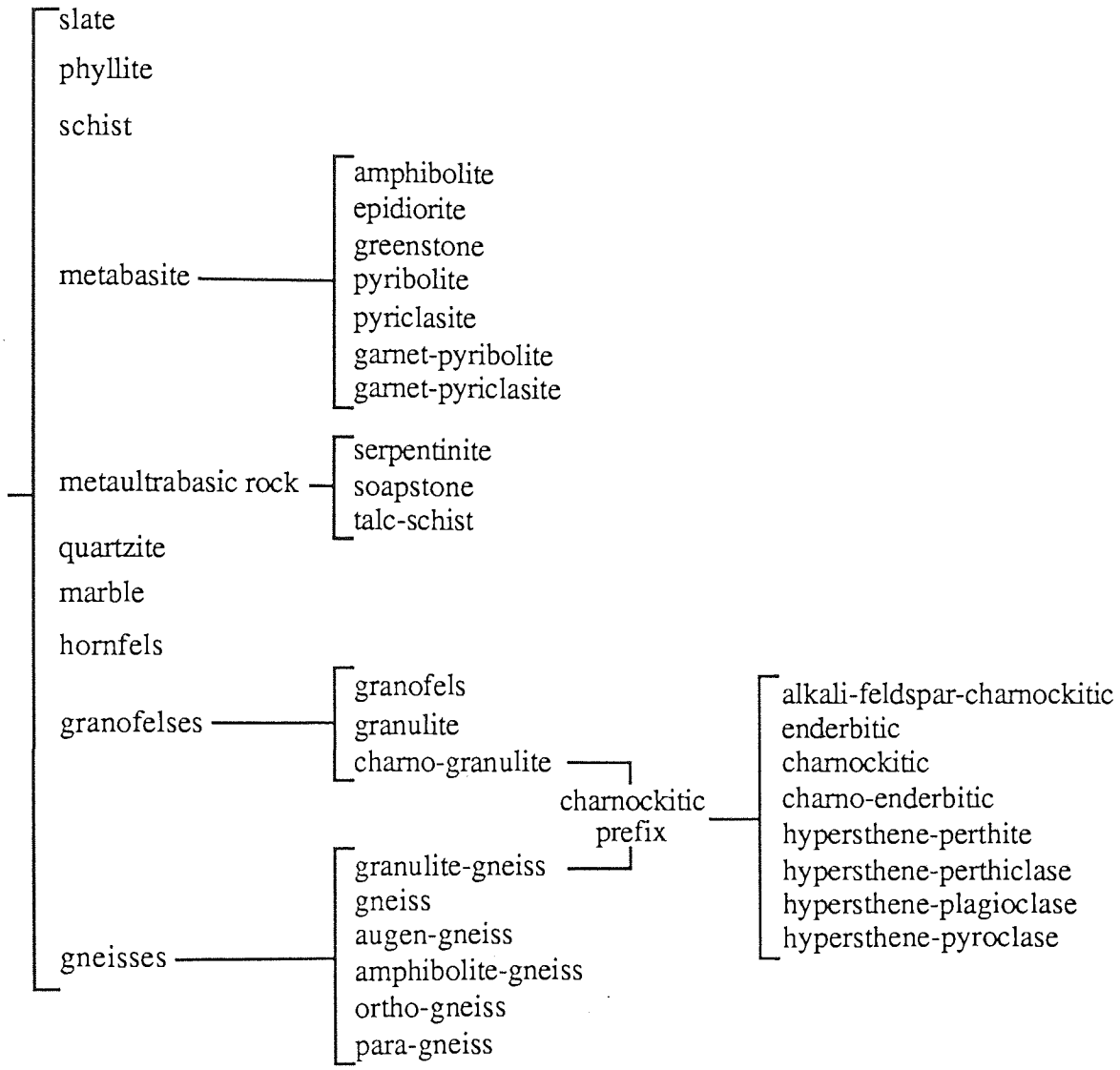

Figure 6.4 Complete 'general name' classification tree as currently implemented in the METEX knowledge base.

METEX is capable of selecting from a variety of different names for the rock, some more restricted in their use than others. The main groups and the available subclasses, wherever applicable, are shown in Figure 6.4. The definitions used for these groups have been taken from the major textbooks used in teaching metamorphic petrography (e.g. Best 1982, Fry 1984, Jackson 1970, Mason 1978, Suk 1983, Turner 1980, Winkler 1979), along with the recommendations and suggestions of SCMR. One area of this classification

worthy of particular attention is that of the rocks termed 'granofels'. Granofels are defined here as coarse grained rocks which show an equigranular, granoblastic texture. This context term has been used to separate those rocks termed 'granulites', which should strictly show granulite grade, from others similar in appearance, but of lower grade. Winkler (1979) attempted to have people use the term 'granolite' to cover the same group. SCMR recommend the general term granofels where the texture is appropriate, but in METEX it has been taken to account only for non 'granulite' rocks. This choice was prompted by the widespread use of the term 'granulite' as a general rock name, and Winkler's failure to have that altered. Here the rocks showing regional hypersthene grade metamorphism are termed 'granulites' as used widely. Those which have the correct composition are further qualified as 'charnockitic granulites' by assessing the appropriate prefix term using the quartz, alkali-feldspar, plagioclase (QAP) composition plot (Winkler 1979) of the rock as shown in Figure 6.5.

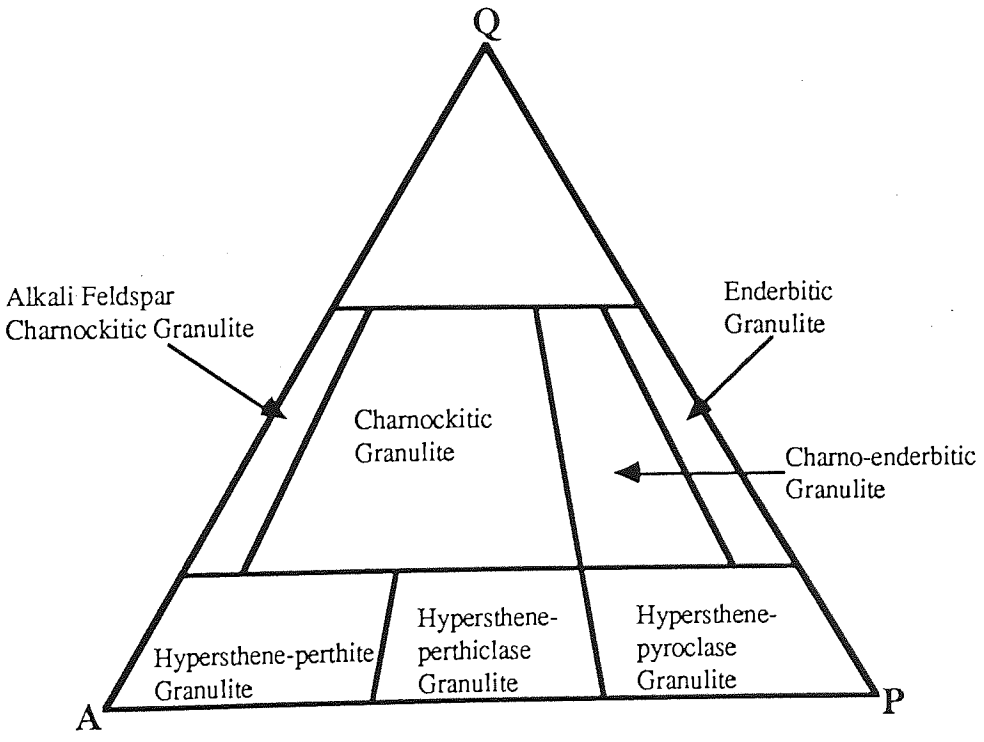

Figure 6.5 QAP ternary plot showing the subclassification of charnockites as used in the METEX knowledge base. (Q=Quartz,A=Alkali-feldspar,P=Plagioclase) (After Winkler 1979)

6.3.2 Precursor Type

Every metamorphic rock, by definition, has been formed by the alteration, or metamorphism, of a rock with completely different characteristics. The metamorphosed rock type, or precursor rock type, is very important to the geologist. Normally, when a suite of metamorphic rocks is examined it is in an attempt to reconstruct their history. The first part of the problem is to identify what types of rock have been metamorphosed, and from that information attempt to reconstruct their original geological environment. This is in reality a very difficult task as the majority of metamorphic rocks are formed by regional metamorphism accompanied by extensive deformation. In such rocks it is not possible to find continuations of the unaltered original rock type outside the boundaries of metamorphism, as we might with say contact metamorphic rocks. This means that we must examine rock specimens very closely to find any information from composition and texture along with rock associations. In many cases only compositional information is of any use since the effects of metamorphism have completely obliterated any relict textures which might have been used. However, even this can lead to problems since:

1. It is possible that rocks from different genetic environments can show similar modal compositions after metamorphism, for example argillaceous sediments can have compositions similar to basic igneous rocks, and arenites similar to acid igneous rocks.

2. It is unusual to find metamorphism which has not been accompanied by fluid activity, normally H_2O or CO_2 fluids, so that the system can not be considered as closed. This means that the chemical composition of a metamorphic rock, which influences and is reflected in the modal composition, can be quite different to the original rock composition.

To solve the problem of ambiguity in compositions of rocks it is necessary to use a combination of relict textures, composition and rock associations which normally require to be reconstructed by removing the effects of deformation. Many geologists are now

using detailed examinations of trace element geochemistry and isotope geochemistry to attempt to resolve problem cases where even the genetic type of the rock is difficult to ascertain (Pierce and Cann 1973). Such examinations do work assuming, as the techniques do, that the trace elements used remain immobile during metamorphism, that is they represent a closed system retaining fingerprints from the original rock. Unfortunately such techniques are out of the scope of this project since the targeted user is unlikely to be able to provide the necessary geochemical data.

The hierarchical tree of precursor rock groups recognized by METEX is shown in Figure 6.6.

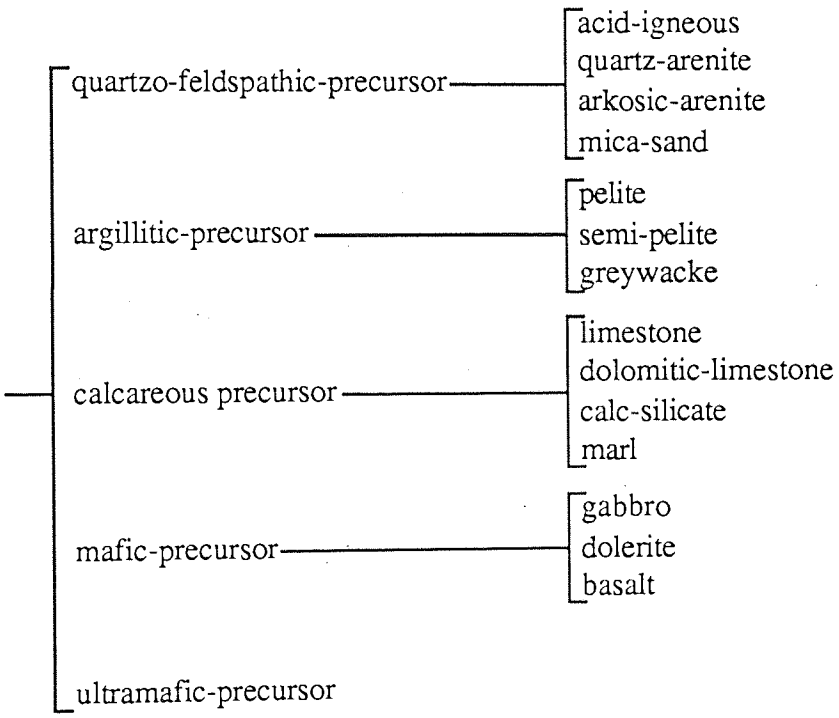

Figure 6.6 Complete 'precursor' classification tree as currently implemented in the METEX knowledge base.

6.3.3 Metamorphic Environment

An important feature of metamorphic rocks is the environment in which metamorphism occurred, sometimes called the type of metamorphism. There are several such environments which are recognized by the geologist. The most common are contact

metamorphism, occurring at low pressure and high temperature at or close to the contact between an igneous intrusion and the surrounding country rock, and regional metamorphism, occurring at higher pressure but similar temperatures during the deformation of deeply buried sediments during orogenic events. Other less common environments are:

1. Metasomatic, metamorphism through high temperature fluids adding and subtracting different chemical phases.

2. Burial metamorphism, occurring as rocks are subjected to increased pressure and temperature as they are buried to greater depths in the crust. Mineralogical changes occur in the absence of igneous or tectonic effects, with temperature increasing as a result of the geothermal gradient.

3. Cataclastic, or dynamic metamorphism, occurring at shear zones where rocks are heated and pressurized by the friction during shearing.

4. Seafloor metamorphism, occurring at low temperatures and pressures on the ocean floor and often accompanied by fluid movement.

5. High pressure metamorphism occurring at subduction zones.

In order to provide both a testbed for applicability of the expert system designed in this project, and also a working system which has some value to the geologist, only the most common regional and contact environments are handled by METEX. The rocks which occur in these environments are covered in reasonable detail and provide enough material to demonstrate the features and performance of the system design and approach. Unfortunately limiting the system to handling only these environments is likely to introduce more errors in its operation. This is because the system will attempt to force all rocks into one of these two basic categories. One additional possibility has been provided in METEX to handle situations where contact metamorphism has overprinted an earlier regional event. It is also possible to find situations where an early contact aureol has been

regionally metamorphosed in a later event. However, under normal conditions it has been assumed that any record of the contact event will have been obliterated during the regional event. It would also be reasonable to expect that the igneous intrusion responsible for contact metamorphism would itself have undergone metamorphism and deformation during regional metamorphism, adding to the problem of recognizing the contact event. In METEX it is assumed that in situations where both environments are shown, the contact event is later than the regional event and overprints it. Also, in cases where both environments are shown, and two grades are also shown, it is assumed that the lower of the grades has been caused by the contact event. This assumption is based on the idea that higher grades of contact metamorphism would obliterate the record of regional grade on a local specimen scale. In order to accurately interpret these rocks it is actually necessary to have information about grades in the regional rocks to properly indicate the association of environment and grade.

6.3.4 Metamorphic Facies

The facies classification of a metamorphic rock is a useful system for describing the P and T conditions under which the metamorphism occurred. Unfortunately, as with the general name, many schemes exist for this task, although most geologists use only one of them. The usual scheme to use seems to be that of Turner (1980) and the subclasses of Turner are shown in Figure 6.7 as fields on a P T graph. METEX uses these subclasses as they provide simple descriptive groupings, although the current trend is to quote P and T estimates rather than subfacies. As stated, Turner's is only one of the schemes used for classifying metamorphic grade. For comparison Figure 6.8 shows Turner's fields along with those of Hietenan (1967), as well as the equivalent fields from the zone concept of metamorphic grade. In METEX the grade of metamorphism is give on Turner's scheme and the equivalent Hietenan and zone fields are shown as extended details.

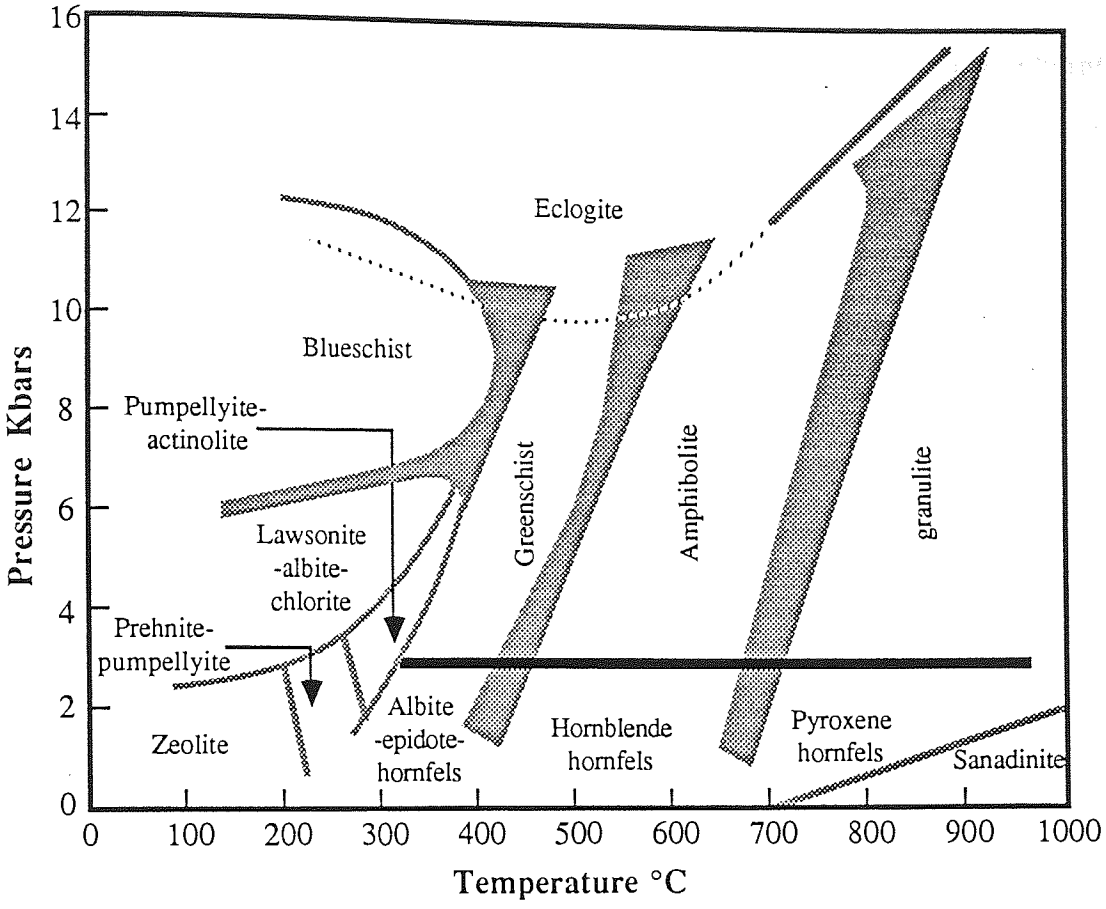

Figure 6.7 Schematic diagram showing relationship of common facies terms to pressure and temperature. After Turner (1981).

Zone concept terms	Chlorite zone	Biotite zone	Garnet zone	Staurolite zone	Kyanite zone	Sillimanite zone	*Hypersthene zone*
Facies terms (Turner)	Greenschist facies			Almandine-amphibolite facies			Granulite facies
Subfacies	Muscovite chlorite	Muscovite biotite	Almandine epidote	Staurolite almandine	Kyanite almandine	Sillimanite almandine muscovite	Sillimanite almandine orthoclase
Facies terms (Hietanen)	Greenschist		Epidote-amphibolite		Amphibolite facies		Granulite facies
Subfacies	Muscovite chlorite	Muscovite biotite	Biotite almandine	Staurolite almandine	Staurolite kyanite	Kyanite almandine	Sillimanite muscovite	Sillimanite Kspar

Figure 6.8 Comparison of the subfacies terms as used by Turner and Hietenan both shown against the zone concept. (After Jackson 1970)

METEX, in an attempt to model the expert, is capable of handling rocks with complex histories. Many metamorphic rocks show very simple histories where only one grade is shown and only one metamorphic environment exists, for example a low grade

regional rock - 'greenschist facies', or a low grade contact rock - 'albite-epidote hornfels facies'. However, it is very common to find rocks which show either progressive or retrogressive metamorphism or even metamorphism at a similar grade but in a different environment. For example we might have a rock which is on the boundary between low and medium grade. This rock could be showing a change from low to medium grade - progressive metamorphism, a change from medium to low grade - retrogressive metamorphism, or low grade contact metamorphism overprinting medium grade regional metamorphism. METEX assumes that where contact and regional environments are indicated, the contact metamorphism is of lower grade and later in time than the regional metamorphism. This assumption is based on the idea that regional metamorphism of any grade, if it is later than the contact metamorphism, will completely re-equilibrate the rock and overwrite the contact mineral assemblage. Also, if the contact metamorphism were of higher grade than the regional metamorphism then it is most likely that, within the limits of the specimen under consideration, the contact metamorphism would completely wipe out any evidence of the regional metamorphism.

In METEX the first thing which is decided is which grade or combination of grades is shown by the rock. This is done by looking at the minerals present in the rock along with the precursor rock type to decide which minerals are useful grade indicators and what grade they represent. Once the grade combination has been found METEX attempts to find the setting of metamorphism, that is, 'regional setting', 'contact setting', 'contact and regional setting'. Each setting in each grade combination represents a context which requires slightly different goals to be solved to construct a suitable facies description. For example in a 'regional low grade' context, only the regional low facies needs to be given, either 'chlorite-muscovite subfacies', 'biotite muscovite subfacies', 'almandine biotite subfacies' or simply 'greenschist facies' if the others cannot be identified. In the case of say a 'contact and regional, low and medium grade' context we would discover the medium grade regional class, the low grade contact class and then form the descriptive

term '<contact class> overprinting <regional class>'. A section of the possible facies classification tree is shown in Figure 6.9 to clarify the possibilities available to METEX.

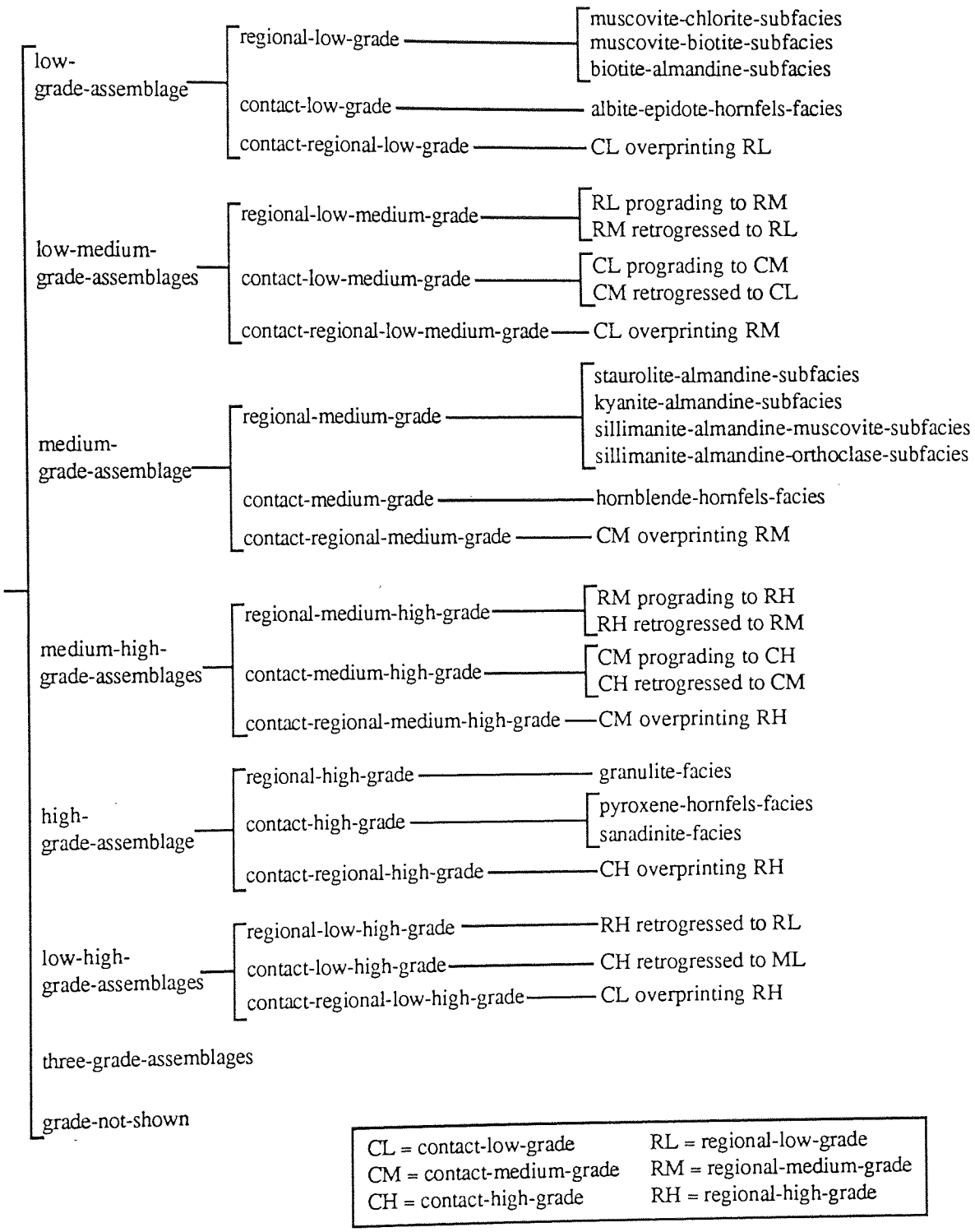

Figure 6.9 Complete 'precursor' classification tree as currently implemented in the METEX knowledge base.

CHAPTER 7

Evaluation

7.1 Evaluation

The evaluation of expert systems which are described in the literature is an area which receives very little coverage by authors. Many systems are described as being successful with little or no reference as to who considers the system successful. However, it must be accepted that the testing and evaluation of an expert system is going to be an inherently difficult task. This is because several distinct areas of performance must be tested simultaneously.

Firstly the performance of the interpreter must be considered. Given some initial data does the interpreter behave as would be expected? For example, we might set up some set of test rules which we expect to produce particular behaviour when uncertain evidence interacts. Given such a set of rules it should be possible to demonstrate that the resulting uncertainties in hypotheses show both the correct values and also complete consistency as the uncertainty in evidence changes. A further area in which the interpreter should be monitored is in determining whether any 'bugs' remain in the design. With expert systems it can be difficult to ensure that all 'bugs' are removed as some combination of data from the real world may never have occurred in initial testing.

Secondly the performance of the complete knowledge base interpreter system must be tested in some way. Again this can be a difficult task since it is impossible to test every possible combination of evidence through the system in order to check integrity. However, it is possible to test system performance against that which would be expected. This testing can only be done by the expert who must produce examples for the system and then check the system's results against this/her own. This approach has been taken by Gaschnig (1982) in his paper on validating the PROSPECTOR knowledge base where the level of detail even compares the degree of probability achieved by the system with that

reached by the expert. This testing was carried out using the designer of the tested model as the expert, which produces a direct test of the system performance. However, it would appear to be of value to have the system tested against other experts. In so doing it should be possible to highlight areas of disagreement which require a greater degree of detail in the knowledge base as well as indicating gaps in the first expert's knowledge. This should make the system more useful as an expert system as it would embody a broader expertise in a particular field than any individual human expert, a fine target even if it is very difficult to attain.

The third and final area in which an expert system ought to be evaluated is in its potential value to the end user. That is, some indication must be gained from both expert and user as to how valuable the system might be in real life use. This is an important level of evaluation since it can be argued that systems such as MYCIN for example have shown high levels of expertise in their domain yet have not gained acceptance by their targeted end user. This may be due to a combination of several factors, broadly covered by dissatisfaction with system performance and use, or concern over legal repercussions in the case of system error, a particular problem for medical systems.

In the case of METEX the evaluation carried out has followed, to a certain extent, the evaluation of PROSPECTOR. That is the system has been tested by the expert against a set of examples for which he expected particular results. It is not intended to enumerate the results, since only limited testing has been carried out within the confines of this project, but rather the results will be discussed in general terms indicating:

1. The degree of accuracy of the knowledge base against the expert.

2. The potential value of the system as a teaching tool for the undergraduate geologist.

3. The applicability of the technique, as used in METEX, to the igneous and sedimentary rock divisions.

The testing was carried out by the expert involved in knowledge base design, Dr J. Ashworth, followed by further testing by postgraduate students and staff at Aston and by an expert from outside the department. Each of the people evaluating the system was asked firstly to test the performance of the knowledge base by giving example cases and noting the accuracy of the METEX result compared to their own. Once the system had been run on a few examples the individuals were asked to consider the value of the current system to the targeted undergraduate user and also to consider the potential of the technique in handling the complete rock classification domain.

The results for this evaluation will be discussed briefly here summarising the salient points. Evaluation by the design expert, Dr Ashworth, produced results for 'general name' and 'precursor rock type' which were consistent in both the classification and degree of belief with those expected for each example. Results gained in the 'facies' classification did not show the same degree of accuracy from the system except in the more simple cases. Once conflicting data occurred, which the expert would handle by considering the timing of polymetamorphic events, the system began to make errors. In the expert's view these problems could be overcome by allowing the system to consider the timing of polymetamorphic events through detailed interrogation, of the user, about the reaction relationships of minerals. This however may present problems for novice users who are likely to produce erroneous data.

Dr. Ashworth was of the overall view that the system performed to a satisfactory, near expert level, within the area of the domain so far encoded. He concluded that the system would serve as a good teaching tool for undergraduates. He also felt that the system might be of use as a research tool for workers outside of the metamorphic domain, mainly because of the use of precise definitions ensuring accuracy and consistency. However, he did feel that the interface required modification to reduce the keyboard work required of the user.

The next stage in the evaluation was to have an independent expert consider the system in order to discover whether or not it had any true value to the geologist. The expert chosen for this task was Dr. A. Wright of Birmingham University. It became clear during Dr. Wright's consideration of the system that the knowledge base as it stands performs to a reasonably acceptable level but does require some fine tuning, by the expert, in order to improve performance. This was clearly demonstrated in examples which the system classified incorrectly. In these, the system and the expert differed in the chosen category boundaries used for compositional terms, i.e. terms such as 'rich' or 'major' used in describing the amount of a mineral phase present. Dr. Wright considered the system to be of some value as a teaching tool, particularly in teaching users the correct strategy in classification and in explaining definitions of rocks via the inference system's explanation facility. This facility however requires some modification in order to have the rule presented in the form given by the expert. As with Dr. Ashworth, Dr. Wright concluded that the system, as it stands, would be of use as a teaching tool for undergraduates in the first and second year. He also agreed that with further development the system may be of use as a first level research tool, particularly with the addition of sedimentary and igneous contexts and a mineral identification sub-system.

The final stage in evaluation of the system was to have experts in the sedimentary, Dr. A. Searle, and igneous, Dr A. Chambers, fields look at the system and consider the potential value in their domains. Both experts here agreed that the system had potential as a teaching tool, and that with careful development in specific areas of their domains the system could become a reasonably useful research tool.

All of the experts who were asked to look at the system came to similar conclusions that a complete petrographic system, based on METEX, would be of some value in teaching geology. In every case it was clear that the expert considered that the system might also have a potential for use as a first level research tool assisting workers expert in one rock division to classify a rock to a reasonably expert level within another division. All of the experts also agreed that the modular, declarative manner in which the knowledge

base could be put together should allow further development to continue in a logical manner. It is the author's personal view that this should allow the system's knowledge to be developed to expert levels by many different experts each specialized in a particular area of geology. If this approach were taken then the system could potentially become a true 'expert system', in terms of its performance.

7.2 Example Session with METEX

An example of one of the test case given to the system by Dr. Ashworth is given below. The example run has been annotated in order to clarify the output. Annotations appear in [square brackets and in a different font.]. The example session shows user input in italics.

WELCOME TO THE KRILL EXPERT SYSTEM TOOL.

This system can run in two forms : SHORT or LONG.

The LONG version explains what is being done at all times. The SHORT version does not give continuous reports. Unfamiliar users may prefer the LONG version the first few times through the system.

(S)hort or (L)ong ?- *l*

[Having asked for a long version of the system the user will be given a detailed account of the system's activity. The system can then activate the 'rock' context frame.]

ROCKS:
It is assumed in using METEX that you have a hand specimen and a thin section for the rock in question. You will be expected to be able to identify the minerals and textural features of the rock whenever they are required. METEX will accept abbreviations for mineral names. If it does not understand an abbreviation or name you will be asked what you wish to do: retype the name or leave it (a good hint here is that if a full mineral name is not

recognized and is spelt correctly then METEX does not use the mineral, so it is probably not important.). You will get better results from METEX if you identify minerals to the highest accuracy you can, e.g. use HYPERSTHENE or AUGITE rather than simply PYROXENE if you can make the required distinction between the pyroxenes.

[The 'rock' context has one task to be solved. This is the 'major division' task, which in this case defaults to 'metamorphic rocks'.]

MAJOR ROCK DIVISION :
All rocks are classified into one of the major divisions of Igneous, Sedimentary and Metamorphic rocks. For the moment we will assume that your rock is metamorphic and we shall attempt to subclassify it as such. However, normally it would be necessary to collect some basic data to make this choice, and since the same data will be required in any case, it may as well be collected now.

[Having introduced the user to the method to be used in classification the system collects the necessary data using pre-defined questions.]

What is the grainsize of the rock?

1 COARSE - >1mm
2 MEDIUM - 0.1 to 1mm
3 FINE - 0.01 to 0.1mm
4 VERY-FINE - <0.01mm

choose a number, or (u)nknown
?- *1*
How certain (0-1).
?- *1*

What is the texture of the rock?

1 CRYSTALLINE
2 CLASTIC

choose a number, or (u)nknown
?- *1*
How certain (0-1).
?- *1*

Is the texture granoblastic?
(y)es, (n)o, or (u)nknown
?- *n*
How certain (0-1).
?- *1*

Does the rock show a foliation fabric?
(y)es, (n)o, or (u)nknown
?- *y*
How certain (0-1).
?- *1*

What type of foliation is shown?
1 FISSILITY
2 BANDING

choose a number, or (u)nknown
?- *2*
How certain (0-1).
?- *0.2*

Please enter the modal composition of the rock in the following format.
<mineral name> <percentage estimate> <return>
Finish with a blank line (press return).
?- *andesine 50*
?- *hyp 25*
?- *di 25*

[The user has typed in an incorrect abbreviation for 'diopside'. The system does not recognize the mineral name. The user is therefore asked to decide upon the next step to be taken.]

Sorry but I do not understand the mineral di. Either I do not know the mineral or you have misstyped it.
Do you wish to retype(r), delete(d) or accept it anyway(a)?- *r*
Please retype the mineral name?- *diopside*
Please continue inputing minerals as above.
?-

[The system now enters the metamorphic rocks context.]

METAMORPHIC ROCKS :
Metamorphic rocks are slightly complex to classify as in general metamorphic petrologists are interested in giving a general name, a precursor rock type, some indication of metamorphic environment, and a facies class indicating metamorphic conditions of P and T. We shall consider each of these different aspects of your sample in turn.

[The first slot, general name can now be activated.]

GENERAL CLASS :
The general class of a metamorphic rock is the name which most geologists would use when describing a particular sample. Some of the more general rock names commonly used can in fact be further sub-classified.

[The system now activates the Eshell inference interpreter to decide between the possible general names. Eshell reports its conclusions and then asks if any explanation is required.]

Probability ranges for general_metamorphic_name are as follows :-
(**meta_basite**) belief = 1.0

Any explanation required for propositions in this context?
yes/y or no/n ?- *y*

[Having asked for the explanation facility the user is now able to ask Eshell about any propositions in the current ruleset.]

?- *what*
The propositions found to be likely are as follows :-
(**meta_basite**) belief = 1.0

?- *what_not*
The propositions found to be negated by the evidence are as follows :-
(**slate**) belief = 0.0
(**phyllite**) belief = 0.0
(**schist**) belief = 0.0
(**gneisses**) belief = 0.0

(**hornfels**) belief = 0.0
(**granofelsic**) belief = 0.0
(**marble**) belief = 0.0
(**meta_ultrabasic_rock**) belief = 0.0
(**quartzite**) belief = 0.0

?- *details meta_basite*

Meta basic rock is a term used here to cover those rocks which show a composition which is predominantly of minerals from the basic association.

?- *why meta_basite*

[In response to a 'why' question the system returns its degrees of belief in the propositions which directly affect the proposition given in the query.]

I believe (**meta_basite**) with belief = 1.0 because :-
I believe (**poly_mineralic**) with belief = 1.0
I believe (**mafic and calc_aluminous minerals are predominant**) with belief = 1.0
I believe (**quartz major**) with belief = 0.0
I believe (**granoblastic**) with belief = 0.0

?- *end*

[The system can now continue its depth first solution of the task 'general name'.]

META-BASITE:
Meta-basic rocks are those formed by metamorphism of basic igneous bodies. They can be the equivalent of gabbros, basalts or dolerites and can be subclassified according to grainsize, fabric, and also composition.

Probability ranges for meta_basic_SC are as follows :-
(**pyriclasite**) belief = 1.0

Any explanation required for propositions in this context?
yes/y or no/n ?- *n*

[The system has finished the task 'general name' and can now return to the 'metamorphic rocks' context. The next slot to be considered is 'precursor rock type'.]

PRECURSOR ROCK TYPE :
By definition, every metamorphic rock has been formed by a process of change from some previous rock type. Rocks can be separated into broad compositional groups which can be further subdivided on the basis of relict textures and field relationships. Here we must first decide upon the compositional group to which a rock belongs.

Probability ranges for precursor_rock_type are as follows :-
(**mafic_precursor**) belief = 1.0

Any explanation required for propositions in this context?
yes/y or no/n ?- *n*

[The system now continues to subclassify the metamorphic precursor.]

To what extent do you believe that : large scale layering is present in the rock body
(answer between 0 and 1) ?- *0*
To what extent do you believe that : the sample is from a small scale rock body
(answer between 0 and 1) ?- *0*

So far my conclusions about this context are as follows :-
(**gabbro**) belief between 0.65 and 0.965
(**basalt**) belief between 0.0 and 0.5
(**dolerite**) belief = 0.45
Do I need to investigate further ?- *n*

[There is little point in trying to further prove 'gabbro' since no other proposition can compete. The system can now report on the task: mafic_precursor_subclass.]

Probability ranges for mafic_p_SC are as follows :-
(**gabbro**) belief between 0.65 and 0.965
(**basalt**) belief between 0.0 and 0.5

Any explanation required for propositions in this context?
yes/y or no/n ?- *n*

[The system can now continue in the 'metamorphic rocks' context. The next task is to classify the 'metamorphic environment'.]

METAMORPHIC ENVIRONMENT :
Every metamorphic rock has been produced by the effects of Pressure and Temperature. This can occur in different environments. The main metamorphic environments are Contact and Regional. Other environments do exist but tend to be less commonly seen.

[The 'metamorphic environment' ruleset has not been written so the system has been provided with a dummy ruleset which it uses to allow the user to solve the task.]

To what extent do you believe that : suspect regional metamorphism
(answer between 0 and 1) ?- *1*
To what extent do you believe that : suspect contact metamorphism
(answer between 0 and 1) ?- *0*

Probability ranges for metamorphic_environment are as follows :-
(**regional_metamorphic**) belief = 1.0

Any explanation required for propositions in this context?
yes/y or no/n ?- *n*

[The final task in the 'metamorphic rocks' context can now be tackled, 'metamorphic facies'.]

METAMORPHIC FACIES:
The metamorphic facies of a rock is an indication of the likely P and T conditions under which metamorphism took place. The facies terms used here refer to the classes of Turner. Before naming the metamorphic facies we must must attempt to identify whether the specimen shows one or more grades of metamorphism. This can occur when a rock has been equilibrated twice, either by the reducing P and T of regional metamorphism allowing a lower grade to be shown along with the maximum grade, or by contact metamorphism overprinting regional metamorphism.

Probability ranges for general_grade are as follows :-
(**medium_high_grade_assemblages**) belief = 1.0
(**high_grade_assemblage**) belief = 0.97025
(**medium_grade_assemblage**) belief = 0.95125

Any explanation required for propositions in this context?
yes/y or no/n ?- *n*

Probability ranges for M_H_grade_setting are as follows :-
(**regional_medium_high_grade**) belief = 1.0

Any explanation required for propositions in this context?
yes/y or no/n ?- *n*

[The system must solve the medium grade and high grade components along with the sense of the overlap, i.e. is it prograde or retrograde.]

To what extent do you believe that : plag composition is An < 50
(answer between 0 and 1) ?- *1*

Probability ranges for RM_facies are as follows :-
(**almandine_AMPHIBOLITE_facies**) belief = 1.0

Any explanation required for propositions in this context?
yes/y or no/n ?- *n*

Probability ranges for RH_facies are as follows :-
(**GRANULITE_facies**) belief = 1.0

Any explanation required for propositions in this context?
yes/y or no/n ?- *n*

To what extent do you believe that : suspect prograde reaction
(answer between 0 and 1) ?- *1*
To what extent do you believe that : suspect retrograde reaction
(answer between 0 and 1) ?- *0*

Probability ranges for PR_sense are as follows :-
(**prograde**) belief = 1.0

Any explanation required for propositions in this context?
yes/y or no/n ?- *n*

[Once the system has solved all of the tasks required in the 'metamorphic rocks' context it is able to produce a summary of its results.]

GENERAL NAME IS:
Pyriclasite
PRECURSOR ROCK TYPE IS:
Gabbro
METAMORPHIC FACIES IS:
Almandine_AMPHIBOLITE_facies prograding to GRANULITE_facies

CHAPTER 8

Future Work and Conclusions

8.1 Future Work

As has been described throughout this thesis the METEX system, as it currently stands, can only be expected to serve as a demonstration of the value of the expert system technique, and in particular the METEX design, in computerising rock classification. It is to be hoped that the system's operation is of enough interest to the geological world to interest other workers in attempting to add new context frames to the system in order to handle the classification of both igneous and sedimentary rocks. It can then be envisaged that the system's ability to hold large amounts of technical information about rocktypes may be of value to workers outside of mainstream geology. For example it should be possible to develop the system to contain the engineering properties of particular rocks so that an engineer, with a limited geological knowledge might be able to have a rock specimen firstly identified and then have further information relevant to his own needs instantly available. However, in order to make the system more useful in this capacity it would be desirable to have a mineral identification system interfaced to the rock identification system. This should then allow for even less experienced people to use the system. Unfortunately however the level of knowledge required to make even the simplest observations requires some prior geological experience and it would pose very special problems if a system were to attempt to have a complete novice gather its raw data. This problem may be solved at some point in the future when the technology allows for computerised image processing direct from a thin section. Although work is under way in this area at the current time the likely level of performance achievable is the identification of given grains, the system having been told what minerals are present. It is obvious therefore that any possibility of a completely automatic identification system can only be considered a possibility in the far distant future.

The interpreter used in METEX poses several problems which have not been addressed in this project. The main problem is that currently the user is not allowed to alter data during a consultation. The reason for not allowing this is that the truth maintenance required to update the knowledge base in such circumstances presents very complex difficulties. The main problem is that the system may find that justifications for currently active contexts are lost requiring backtracking across several levels of control, possibly with data having been gathered which loses its integrity or which must be recollected within the correct context. The truth maintenance problem is however made more complex in situations where the system infers some fact within a context which is itself justified by a previous belief in that context. In dynamic worlds such situations are accepted because the world itself may have changed. However, in a static world as used in METEX such a situation is totally unacceptable since the system in removing the justification for the context must remove the justification for the removal which stemmed from the context itself. At the moment the knowledge engineer must ensure that such situations cannot occur. Unfortunately this limits the flexibility of the knowledge representation which ought to be able to run truth maintenance. The METEX interpreter has been designed throughout with a truth maintenance system in mind so that although it has not been implemented all of the hooks required for its implementation are available, for example all data maintains a record of its justification so that at any time the justification for any active context can be traced to a particular data point. Should this data point change then the system should be able to trace all subsequent justification changes reliant upon the altered data.

Another area in which work could be considered in the context of the METEX system is in simplifying the knowledge base maintenance. This could be achieved by designing a knowledge acquisition tool which allows for simple input of new contexts, rulesets, questions and semantics. If the input system were designed with the ability to test for inconsistency and for conflict in knowledge then the system would be of immense help to the expert required to ensure that the system remains up to date. This is however not of

real importance in rock classification because, in spite of the imminent guidelines for metamorphic classification, it is unlikely that definitions will be altered very regularly. It would also be of some value to have such an acquisition tool create separate files for knowledge sources and to only load these into memory as they are required. This is not important in systems with small knowledge bases, but as more knowledge is added memory considerations are likely to become more important.

8.2 Conclusions

METEX has demonstrated that the expert system approach to classification is capable of handling the domain of metamorphic petrography. This allows us the advantage of an easily maintainable system for which alterations to definitions are relatively simple, an important factor in a domain where work to develop a systematic classification scheme is still ongoing. The system as it stands has been designed with the undergraduate user in mind providing both classification assistance and additional information - in many ways simulating an electronic textbook. Evaluation of the system by both experts and postgraduate students has revealed that the system should have some value to undergraduate students as a tuition aid. The experts agreed that the system as it stands demonstrates the value of the technique in providing such a tool for students. However, all of the experts who looked at the system were of the opinion that it could serve some purpose as a first level research tool if the knowledge base were to be developed to handle classification of all rock types to an expert level. This development would require a considerable number of man hours of work but could be achieved by using the modularity of the knowledge representation to allow stepwise development. In utilising a stepwise approach there is great potential for the system to be made very expert in terms of its performance. This is because as the system reaches the stage of classifying a rock into a particular context, if that context allows further subclassification then a new expert in that context could be used to develop the ruleset required for the subclassification.

The interpreter used in METEX has been purpose designed to handle metamorphic classification. However, the strategy has always been that the interpreter and knowledge base should be easily separable. It is not clear whether it would be found that this particular knowledge representation would be of any value in other domains but it is to be hoped that it might. The modularised approach provided by the use of context frames and segmented rulesets allows for a logical, incremental growth of knowledge bases. This should allow for simpler testing since each level of a classification can be thoroughly tested before subsequent levels are added. Even in domains where the need for multiple task handling does not exist the interpreter would still be of some value in partitioning knowledge. This is particularly important in classification where the expert tends to provide different levels of classification, modelled by different levels of context frame. When the expert is unable to complete a subclassification he is still able to give a less detailed classification from the previous level. For example it may not be possible to subclassify a mafic precursor but the expert may still be sure that the rock *has* a mafic precursor. It would be very difficult to handle this in the normal representations used for classification.

The final area of the interpreter design which is of special interest is the method used to handle uncertainty in inference. The method used in METEX (described in Chapter 4) is very much original since it combines features of several well documented uncertainty systems, namely MYCIN, Dempster-Shafer and Eshell. The system can be shown to be well founded in the Dempster-Shafer theory in as much as Gordon & Shortliffe (1984, 1985) have shown that the MYCIN CF system is a special case of Dempster-Shafer theory. The system uses a two value range system to represent uncertainty. This allows the interpreter to handle negative inference in a consistent manner whilst also allowing the expert to distinguish between 'necessary' and 'sufficient' evidence in inference. The range system of representation also allows the alpha-beta based, question selection algorithms of Mellish (1985) to be used in order to limit direct questioning of the user, by the interpreter, to a minimum. The system has been verified against the INFERNO system of Quinlan

(1983). This verification revealed that while the two systems produced very similar results the precise values were different. This was caused by the use of MYCIN type evidence combination in METEX, which it could be argued is better than the INFERNO min-max combination system. Certainly the METEX system is more intuitive in that it seems sensible to increase the degree of belief in a hypothesis, when many sources of evidence point to the truth of that hypothesis, above the degree of belief from any one of the evidence sources.

APPENDIX 1

User Manual

A1 Using Metex for Rock Identification

This user manual describes how to activate the METEX system from any user space on the Aston University Computer Services' VAX Cluster. If METEX is to be used on any other installation then the initialisation command files described in Section A1.2 will have different addresses from those given here. In such circumstances users will require to contact their own system managers to find out what the addresses are and to ensure that the files contained there have been set up correctly. Please note that whenever a command is illustrated in this document, system prompts are given in bold typeface and user input is given in normal typeface.

A1.1 System Access Requirements

In order to use METEX as described in this documentation the user will first require to have access to a valid USERNAME on the VAX Cluster. If a USERNAME is not available then you should contact Computer Service Enquiries.

A1.2 Hardware Requirements

METEX has been designed to run on a VT100 terminal or any good VT100 compatible terminal. This requirement is brought about by the use of VT100 control codes which will not run correctly on any other terminal type. Unfortunately this also applies to some VT100 emulators where some of the controls do not operate. If in doubt as to whether any terminal is suitable for use with METEX, the best test is to attempt a run of the system. If the first few screens do not look as though they are correct then abort the run using the method given in Section A1.5.1.

A1.3 Setting up to use METEX

In order to make METEX available for use, some initial work must be done to set up the commands necessary to run the system. Users should first logon to the Cluster in the normal way. If you are unsure about how to do this then please refer to the VAX user manual which describes clearly what you should do. (If users have difficulty getting access to the Cluster machine then either your Departmental Computer Officer or Computer Service Advisory should be contacted.)

Once logged onto the system, the first step should be to copy the METEX commands file, METEX.COM, into your filespace. This is done with the following command:

```
$ COPY $1$DUA3:[DORANSH.COMMANDS]METEX.COM METEX.COM
```

A new file METEX.COM will be generated in your filespace. This file sets up the POPLOG system and initialises your filespace to load METEX. Please note that the copy command above is only required when no copy of METEX.COM is present in your filespace. So unless you delete the file it should not require to be copied again.

Now and every time you logon to the Cluster you will require to run this file if you wish to use METEX. This is done by typing the following command to the Cluster:

```
$ @METEX
```

Users more familiar with VMS may wish to add this command to the LOGIN.COM file to save having to type it. This should be done following the documentation in the VMS user guide.

A.4 Running METEX

Once the file, METEX.COM, has been run you are ready to begin using METEX. To start the system you should type the following command:

$ pop11/metex

This command will start the POPLOG system in the language POP11 and load the METEX programs into memory. The computer will print the message

===> <true>

You should now run the METEX system by typing the command:

: metex

A new screen should now appear similar to that shown in Figure A1.1, and you should be able to run a complete session with METEX following only the instructions given on the screen. However, some useful points will be covered in the remainder of this document and should be read at some point.

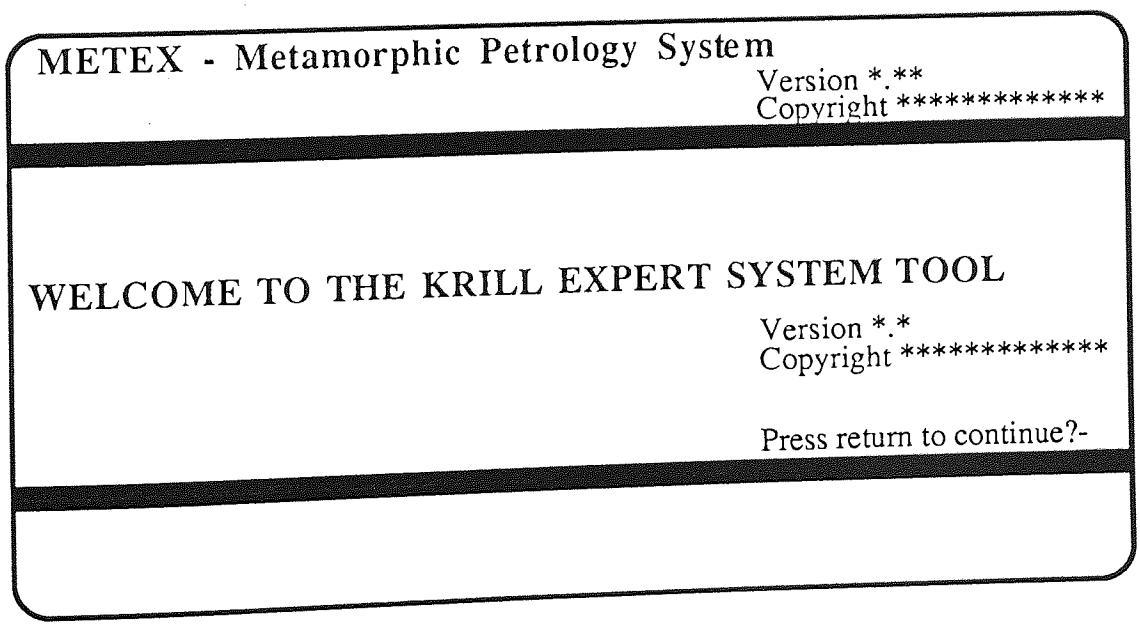

Figure A1.1 Initial screen presented to the user on successfully loading and running METEX.

A1.4.1 The METEX Screens

In METEX the screen is split into three independent sections as shown in Figure A1.2. By splitting the screen in this way it is possible for METEX to give more than one piece of information to the user. Window 1 is normally used to identify the system as METEX and to give the version number of the system. Window 2 is the dialogue window through which all communication between the user and METEX is carried out. Window 3 is a help window which METEX uses to give added information to the user whenever this is likely to be required. For example this window might be used to clarify what sort of response is required in a particular question.

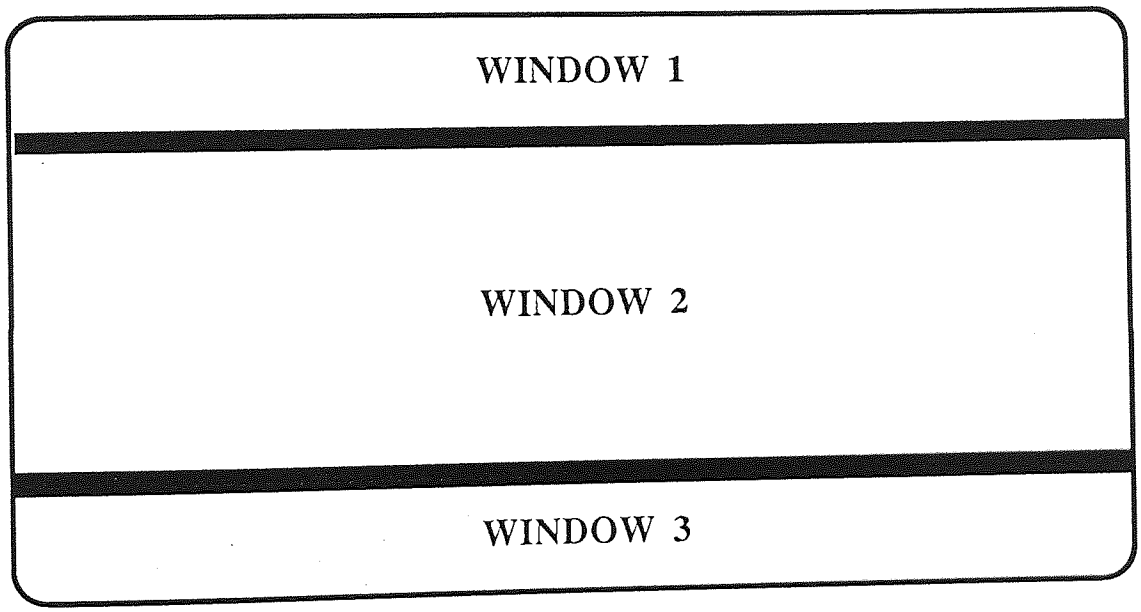

Figure A1.2 Example METEX screen giving window names as used in the text.

A1.4.2 Answering System Questions

METEX will ask various questions while attempting to classify a metamorphic rock for you. The questions are asked in two distinct sections of the program, and as the user, you should be aware of the differences between the two sections of the program.

In the main control area of the program three different question types exist. You will either be asked:

1. To give a simple Yes or No answer to some question, for example 'Is the rock foliated?'. You will then be asked to give some measure of your belief that the answer chosen is correct. This will be on a 0 to 1 scale where 0 indicates complete uncertainty and 1 indicates complete certainty. Please note that in choosing between yes and no you have committed yourself to having some measure of belief that your choice is TRUE.

2. To choose a single entry from a menu of possible values for some property of the rock. For example you might be asked to identify whether the grainsize is 'coarse','medium', 'fine' or 'very fine'. As in the simple Yes/No question above you will be asked to give a measure of your belief that you have given the correct choice of value.

3. To give a list of items according to a clearly defined structure given in the question. This is the type of question used to ask for the modal composition of the rock. You are expected to give, in this case, each mineral followed by its estimated proportion of the rock. In all questions of this type you will be required to press return after each complete entry. To show that you have completed the list you should always finish with a blank entry, achieved by simply pressing the return key. In this type of question no measure of belief will be asked for.

In each of these question types it is possible to ask for help from the system. Since each question should be self explanatory and give clear details about expected responses, the help which will be given in any of these questions will take the form of a more detailed account of the definition of any geological terms, as they are defined within METEX. To ask for help you should simply type 'help' when being asked the question. Please note that many questions will not have any more details available where such extra detail has not been considered necessary. Where this is the case METEX will return a simple apology for not having more information.

The second phase of METEX in which you might be asked questions is during inference. These questions are asked only when the system considers them to be of value in deciding on which of a group of facts might be true. For these questions only one type of response is available, that is, a number indicating your certainty that a propositional phrase is true. However, you may ask for help in these questions and in this case you will be given a list of other valid requests you may make of the system. These requests allow you to interrogate the inference system of METEX to discover why it is asking you a particular question or to discover why a particular proposition is believed. You can also ask for details of a proposition which should give you a more detailed explanation of the meaning of the proposition. The methods used to interrogate the system in this way are explained when you ask for help in this type of question. Please note that you can only ask for explanations in this way when the system is running its inference procedures and not when the control program which asked questions of types 1,2 and 3 above is running.

A1.4.3 Interrogating the System

In many instances you will not be interrogated by the inference procedures at all. You will however be given a chance to interrogate the system about its inference after each time that the inference procedures have been run. In this case the system asks you if you wish to enter an explanation dialogue after it returns its results. If you choose to interrogate the system, then you will be shown a list of possible questions in the help window. There are three basic types of information available to you:

1. "what" and "what_not" simply allow you to have the system show you what goals were found to be possible and what goals were disproved, respectively.

2. "details" when followed by a proposition will access any extra information stored with the proposition.

3."why" when followed by a proposition will give information on what other propositions were used by the system in inferring the requested proposition.

A1.4.4 Reading System Results

METEX returns its results in two stages. As each ruleset is used by the inferencing system you will be given a list of all of the goals found to be likely. Each proposition shown is enclosed in brackets in order to clearly identify the wording of the proposition. A numerical value is also given indicating the degree of certainty with which the system believes the proposition to be true. A 0 here indicates that the proposition is unsupported and a 1 indicates complete support. If two numbers are given then some uncertainty about the truth of the proposition remains, and the degree of support lies between the two numbers. The results given at this stage are based on the assumption that METEX has been investigating hypotheses within the correct context and are therefore not necessarily going to be the final results. For example, METEX may decide that a mistake has been made, and go on to alter an earlier choice ignoring any work done since that choice was made.

At the end of a session METEX will give a summary of the best results given within each of the important areas of the investigation. These results will indicate the hypotheses with which METEX can be reasonably satisfied, while the intermediate results indicated the certainties of the various hypotheses within their context.

A1.5 Ending a METEX Run

METEX may be stopped from running in one of two ways. Section A1.5.1 indicates the graceful manner in which users should at least attempt to exit the system. However occasionally you, as a user, may require to quit the system in mid session. If this is the case then the correct procedure for doing this is shown in Section A1.5.2.

A1.5.1 During a Run

If you require to stop METEX in a mid run this is possible by first escaping from the program. This is accomplished by pressing <control> and <C> keys together. You should now get the following message:

```
SETPOP
:
```

You should now reset the terminal to remove the METEX windows by typing 'termreset' followed by return. The screen will clear and the colon prompt appear at the top of the display. Now press <control> and <z> together to exit to VMS.

A1.5.2 After a Run has been Completed

After METEX has finished running on a sample you will be asked if you wish to continue with a new sample or to finish with METEX. If you indicate a desire to finish then METEX will automatically return you to VMS.

APPENDIX 2

Knowledge Engineering Manual

A2.1 Creating or Altering a Knowledge Base

METEX exists as two quite distinct units, the domain specific, metamorphic knowledge base and the more generally applicable knowledge interpreter and loader known as KRILL. The routines used in KRILL have been designed purposely to be independent of the metamorphic petrography domain so that it could potentially be used as a development tool in other domains. The tool would only be of any special value in domains where multiple tasks are required to be handled in a hierarchically structured tree of contexts. In such domains the full control system of KRILL could be used as it is in the METEX system. This appendix is intended as a short reference for anyone who wishes to use the KRILL program for development of a new knowledge base, or for anyone intending to make modifications to the METEX knowledge base. The permitted syntax used in defining control knowledge and rules is described along with details on how to compile and debug a knowledge base. Where necessary examples of actual definitions will be given from the METEX knowledge base in order to illustrate syntactic definitions. Anyone working with the KRILL tool should be aware that a knowledge of the POP11 programming language is necessary if major development or alteration is envisaged. The use of POP11 in METEX adds much to the system's power and to the flexibility of the development tool.

A2.2 Basic Knowledge Structures and Syntax

METEX uses four basic types of knowledge structure which are recognized by the interpreter. These are as follows:

1. FRAMES - context frames which store the related control knowledge about a particular distinct hierarchical area, or context of knowledge.

2. RULESETS - which store the inference rules related to solving particular problems.

3. SEMANTICS - which are used to represent the relationships, normally simple is_a type relationships, between different basic terms.

4. QUESTIONS - used to represent questions which can be put to the user in order to gather basic important data. These are the main method by which METEX gathers data, although the inference interpreter can ask the user about propositions which it is unable to infer.

A2.2.1 Frames - Use and Syntax

Frames are used to store control knowledge relevant to a particular context within the overall context tree of a problem domain. Frames store the individual tasks which must be tackled in order to satisfy the solution of the frame itself. These tasks are represented in the slot entries of the frame. The syntax for defining context frames is as follows:

```
FRAME <frame_id>
      VIA <var_name 1> ... <var_name n> ; (variables leading
                                          to a frame being activated.)
      INTRO <'string 1' ... 'string n'> ; (text to be used as an
                                          introduction to the
                                          context.)

      SLOTS
          <slot 1>
             :
          <slot n>
      ENDSLOTS
      DEMONS
          <demon 1>
             :
          <demon n>
      ENDDEMONS
ENDFRAME
```

For example:

```
FRAME meta_basite
    INTRO 'META-BASITE:'
        ' '
        'Meta-basic rocks are those formed by metamorphism'
        'of basic igneous rocks. They can be the equivalent of'
        'gabbros, basalts or dolerites and can be subclassified'
        'according to grainsize, fabric, and also composition.';
    VIA general_metamorphic_name;
    SLOTS
        SLOT meta_basic_subclass
                ASSIGNS meta_basic_SC
                FROM meta_basic_rules_SC
                CAN_BE [amphibolite
                        epidiorite
                        greenstone
                        greenschist
                        pyribolite
                        pyriclasite
                        garnet_pyribolite
                        garnet_pyriclasite]
        ENDSLOT
    ENDSLOTS
    DEMONS
        D_ D1 WHEN [slot_finished("meta_basic_subclass")]
                DO[make_result("gen_type",meta_basic_SC)];
    ENDDEMONS
ENDFRAME
```

The first entry <frame_id> must be a single, legal POP11 word and will normally be the same as a possible goal in one of the slots of a hierarchically superior context frame. When a slot task chooses this goal as its solution then the <frame_id> is put on the active frame agenda so that the new frame <frame_id> can be activated by the interpreter. If the goal which should cause the frame to be activated does not have the same name then it is still possible to cause the interpreter to activate it by having its name <frame_id> put onto the agenda manually, by using a suitable demon (procedural attachment).

The variable names recorded in the VIA entry are those variables which might be assigned the name of this frame as a result by some higher level slot, and which in effect record the activation of a frame. It is through the use of these variables that the interpreter is able to reason about the value of a frame for backtracking. The variables store a flag which indicates the reliability the system attaches to the value assigned to the variable. It is important to have all of the variables associated with activating a frame recorded here since

the frame must, on activation, be able to discover how much reliability is attached to its activation.

The entry INTRO is a simple entry in the frame used to provide an introduction, for the user, whenever the frame is activated. This introduction is optional and where used should give the user some information as to the intentions of the context frame. The individual strings should not be longer than 80 characters, otherwise they will not appear formatted. Any number of strings can appear as the interpreter will wait until each page is read by the user before printing the next page. No string should extend onto a second line in the knowledge base otherwise the POP11 compiler will issue an error message and compilation will fail.

A2.2.2 Slots and Demons

The control entries in a frame are the slots and the demons. Slots encode task specific control, much of which is implicit, while demons encode explicit control. The order in which demons appear is unimportant as they are all tested until none fire each time they are scanned. However, with slots, the order in which they are written in the frame dictates the order in which they are obeyed, unless a demon fires to elevate a slot in the agenda, normally by causing it to be added to the front of the agenda. The syntax for coding individual slots is as follows:

```
SLOT <slot_name> (any legal POP11 word)
      ASSIGNS <var name> (this name may appear in a frame
                  VIA entry)
      INTRO <string 1> ... <string n> ;
      PREACTS <list of pre actions> (a POP11 list of items to be
                  run before the slot - optional)
      FROM <ruleset name> (a valid POP11 word naming group
                  of inference rules)
      CAN_BE <goal list> (a POP11 list of possible goals -
                  normally POP11 words)
      DEFAULT <default result> (a valid POP11 word -
                  optional)
      DEMONS
          <demon 1>  ...  <demon n>
      ENDDEMONS
ENDSLOT
```

For example

```
SLOT meta_ultrabasic_subclass
        ASSIGNS meta_ultrabasic_SC
        FROM meta_ultrabasic_rules_SC
        CAN_BE [serpentinite
                soapstone
                talc_schist]
ENDSLOT
```

The slot entry ASSIGNS declares the variable in which the slot will store its result. This variable, if the slot can lead to a further frame being activated, should appear in the VIA list of those frames which are able to be activated. The INTRO entry is similar to the INTRO of a frame and serves the same purpose. The syntax here is the same and the same rules about string lengths apply.

The PREACTS entry of the slot records those major control structures which must have been processed before a slot is able to run to a complete solution. Normally this will be a list of question identifiers to enable data collection within its correct context. However, it is possible to have frames called here in order to force completely new context trees to be activated before the current one can continue.

The CAN_BE entry of a slot should appear as a POP11 list containing, as its elements, the possible solutions to the slot. These solutions should normally be POP11 words and should appear as inferable propositions in the ruleset identified by the FROM entry of the SLOT. The ruleset name should be given here, and should again be a valid POP11 word. It is important to ensure that the goals do appear in the ruleset otherwise errors will occur when attempting to run the knowledge base.

Demons appearing in a slot are only able to be run when the slot is active. This limits their use to particular points during the execution of a frame. Throughout the knowledge base demons must follow the same basic syntax whether they appear in a frame, slot or question definition. The syntax for DEMONS is as follows:

```
D_ < id> WHEN [ <valid POP11 procedural expression> ]
         DO   [ <valid POP11 procedural expression> ];
```

For example:

```
D_ DFGG2 WHEN [charnockite_SC /= false]
            DO [charnockite_SC >< '_gneiss' ->
                granulite_gneiss_SC;
                make_result("gen_type", granulite_gneiss_SC)];
```

In the definition of the demons every POP11 expression appearing in the precondition and action lists must be valid POP11. Care must be taken when defining and testing DEMONS as even syntactic errors will not show up until the code list is run by the interpreter. This is because the POP11 code is not compiled until the interpreter asks the POP11 compiler to make use of the code. Care should be taken with I/O in demon code as the compiler will not compile input from the terminal. It is therefore necessary to redefine functions such as readline and listread if they are required. A version of readline called readline1 has been defined for you and works in the same way as readline. Readline1 should satisfy most requirements in normal dialogue since any list it produces can be compiled by 'popval(list)' to return the correctly compiled input. Please note that this does not apply only to code within the actual demon definition but also to code appearing in user defined functions called from within the demon.

A2.2.3 Rulesets - Use and Syntax

Rulesets are used to store together rules of inference which are related. Normally the relationship will be their use in solving a particular set of possible goals to find the value of some variable. A ruleset can potentially be called by more than one slot and the exact set of goals may be different in each case. However, it would be normal to have different rulesets if the goals are completely different. Rulesets should be defined according to the following syntax:

```
E_RULES <ruleset name>  (a valid pop word same as value of
                         FROM entry in an appropriate slot)
    CONCERNS <var 1> ... <var n> ; (variables potentially
                                    set by the result)
    B_K <information about purpose> ;
    PROPOSITIONS
        <proposition 1>
              :
        <proposition n>
    ENDPROPS
ENDE_RULES
```

This structure is self explanatory and the only part which requires further details is the definition of individual propositions. These define the individual node of a semantic net like dependency graph. The nodes store the details about themselves in records. In the knowledge base only a limited amount of data must be given as the interpreter and compiler will fill in details for themselves. The syntax for the information which must be provided is as follows:

```
PROP <proposition name> ; (one or more POP11 words)
    TEXT      < text list 1>
                    : (added details for the user)
              < text list n> ;
    RULES
        <rule 1>
           :
        <rule n>
    ENDRULES
    ASKABLE       } (these two are optional)
    NOT_ASKABLE }
ENDPROP
```

For example:

```
PROP slaty fabric;
    TEXT [the rock shows a planar fabric along which it splits]
         [into regular sheets of about 0.1mm thick.surfaces]
         [will show a faint sheen from small mica grains];
    RULES
        ER_ 2.1 H.Dorans 25/3/87 giving slaty fabric;
                IF foliated
                AND fissile
                AND very fine grained
                THEN [1 -1] slaty fabric;
    ENDRULES
    NOT_ASKABLE
ENDPROP
```

It is proposed that only the syntax of rules will be given here and that anyone using the system will have read Chapter 4 of this thesis. Chapter 4 documents how rules are used in the inference system and how the evidence from them is combined as well as describing how rule weights are used. Rules can be written with either one weight, used only when the premises have a positive measure of belief, or two weights one for positive and one for negative measures of belief. Rules are written according to the following syntax:

ER_ <rule no.> <information string> ;
IF <premise clause>
THEN <wt clause> <conclusion clause> ;

Where <wt clause> is either one weight or a pair of weights in a list, e.g. [wt1 wt2]. <conclusion clause> should be the same as the proposition name. The <premise clause> of a rule should be formed as follows:

<prem 1> comb <prem 2> ..... comb <prem n>

where comb can be any of AND, OR, AND NOT, OR NOT. The highest priority combinator is OR so that an expression like 'a OR b AND c' means the same as 'a OR |* b AND c *|'. The symbols |* and *| are used to bracket expressions to remove any possibility of ambiguity for the reader of the knowledge base.

A2.2.4 Semantic Relation Rules

METEX represents the hierarchical interrelationships of mineral species in a simple IS_A type semantic net. By making use of such a representation it is possible for the expert to express knowledge using information from different levels in the hierarchy. This gives a certain freedom when writing rules and ensures that the system makes sense to the user when running. For example, we may wish to write a rule which is interested in whether a rock contains mafic silicate minerals. Using the net we can represent this node

as containing pyroxene, amphibole and olivine, whilst under pyroxene we find OPX and CPX with augite and diopside below CPX. Now if the user were to input diopside as a constituent of a rock then the system would realize that a mafic silicate was present. This means that a rule can be more meaningful since it is encoded at the level of abstraction required by the expert.

Nets are encoded in blocks of relationship rules, all rules within a block being relevant to that net. To declare a net the following syntax is used:

SEMANTIC <net name>

rules

ENDSEMANTIC

Each link in the net is declared within this block with no order restrictions as the interpreter will build the net from simple relationships. However for ease of maintenance it is suggested that the links are grouped into conceptually meaningful clusters. Each relationship is declared as follows:

SR_ min_1 ==-> min_2;

Where min_1 and min_2 are single POP11 words. Using this syntax the example net around mafic silicates can be produced:

```
SR_ augite ==-> CPX;
SR_ diopside ==-> CPX;
SR_ CPX ==-> pyroxene;
SR_ OPX ==-> pyroxene;
SR_ pyroxene ==-> mafic_silicate;
SR_ amphibole ==-> mafic_silicate;
```

Accessing Nets

The internal representation of nets is extremely simple and the interpreter only handles the construction and update of the nets. In order to access data stored there the knowledge engineer must construct access functions in POP11 to find the data they require. This makes the use of the net slightly more complex but allows the whole

construct to be much more flexible. The necessary functions once written are accessed via procedural attachments on individual propositions either individually defined or defined as proposition templates. For example in METEX the user is asked to give a list of the minerals contained in a rock sample along with their percentages. If the expert then requires to discover if a particular node is a major component, say CPX for instance, then a function checks to discover if the total of CPX is high enough to be considered as major. This is achieved by finding the CPX node in the net then checking if it is 'in', i.e. proven true. If it is not 'in' then the total is 0%, otherwise the total is found either directly from the user's input or by recursively totaling the daughter nodes to CPX. In order to activate this process a general template is provided which will check any node required.

```
PATTERN ?min_node_X is a major component;
        TEXT [The mineral ^min_node_X is a major component]
                [of the rock.];
        EXP [mineral_total("^min_node_X") @ major];
ENDPATTERN
```

The test in this template checks the percentage of the required node against the 'fuzzy set' associated with the term major. This check is carried out by the '@' operator which returns the certainty with which the calculated percentage falls within the boundaries of the term 'major', see Section 6.2.

Multiple Nets

There is no restriction upon the number of nets used in a single knowledge base provided that the access functions are provided where necessary. Each net is instantiated by firing a procedural attachment within the question construct which is used to find the necessary data for the net. In this way a net will only be instantiated if the data for a specific case indicates that the net is required (see Section A2.2.5.4 on Data Directed Questioning). Instantiation is achieved by using a demon structure within a question definition. Although the process is relatively simple a few things must be remembered. An example of net instantiation from METEX follows:

```
D_ Q4_1 WHEN [mode.var_stat /= "UNASKED"]
        DO [for x in hd(mode.var_val) do
                X(1) :: sem_new_facts -> sem_new_facts
            endfor;
            add_agenda("mineral")];
```

This demon checks that the mode (mineral composition) has been asked for and then sets up a global variable for the semantic interpreter. The global variable 'sem_new_facts' must be set up with the new raw data for the required net before the net is activated, as it is this list which triggers the initial nodes in the net hierarchy for further propagation. Once this variable is initialised the required net can be put on the agenda for activation, the net in this case being named "mineral".

A2.2.5 Question Definitions

It is possible to have data collected from the user via two distinct paths.

1. The expert defines specific questions which can be asked at pre-defined stages in the operation of the system and only asked if the controlling context has been activated.

2. The inference system finds that some proposition is either defined as askable or has not been defined at all (used in a rule but not defined). In this case the inference system will decide if asking about the proposition is likely to influence the goals of the inference process. This ensures that the question is only asked firstly if the correct context is active and secondly if the reply is likely to affect the outcome of the inference.

It is only proposed to consider questions of type 1 here since these are used at specified points in a consultation and can be written in order to cause a data directed line of questioning which will appear to be relatively intelligent to the user. It should be noted that questions which are pre-defined as described here will not be asked if the activating context is not itself activated. Questions are defined in a separate block within the

knowledge base. Questions which the knowledge engineer feels may be of value in the future can be pre-defined and need not be referenced by any current knowledge structure. However, any question which is referenced by a current knowledge structure must be defined. Three basic types of pre-defined questions exist and these are described separately in the following sections.

Simple Questions Type

Questions defined as being 'simple' are questions requiring Yes/No answers to assign a boolean value to some attribute, although a measure of the user's uncertainty can be attached. The user will be asked to reply to the question by indicating whether the attribute is believed true or false and then with what measure of belief the chosen value is to be used. This information is used to build a data structure similar to a MYCIN <attribute> <value> <tally> triple. Questions are defined according to the following syntax:

```
QUESTION <Identifier>
      TYPE simple
      TEXT ['A list of strings to be used as question text.']
      EXPECT ['(Y)es, (N)o, (U)nknown.']
      ASSIGNS <var_name>
      HELP 'A set of strings for use as help text if required.';
      B_K <book keeping details>;
      DEMONS
          :
      ENDDEMONS
ENDQUESTION
```

For example:

```
QUESTION Q3
      TYPE simple
      TEXT ['Does the rock show a foliation fabric?']
      EXPECT ['(y)es, (n)o, or (u)nknown']
      ASSIGNS foliated
      DEMONS
          D_ Q3_1 WHEN [predicate_checks ("likely", foliated,
                                          "yes")]
                  DO  [add_agenda("Q6")];
      ENDDEMONS
      B_K    'to find fabric -- H Dorans 3/87'
ENDQUESTION
```

The identifier of the question must be unique in order that the system can correctly follow references to this structure. Once the system activates the question interpreter with a question the user will be presented with the text strings in the TEXT entry of the definition followed by the EXPECT entry as a prompt. When the user responds to the question then a record will be set up containing the result and will be stored in the variable named in the ASSIGNS entry. This variable will become global so that it can be referenced by any part of the system. The structure of the variable will be as follows:

var_val	var_source	var_stat
[[yes [lb ub]] [no [lb ub]]]	<Question ident>	ACTIVE

In the case of simple questions the var_val entry will in actual fact contain an entry for both the 'yes' and 'no' entries. The user's chosen value will assume lower and upper boundaries as given by the user. If the CF given is not equal to 1 then any remaining certainty is attached to the alternative response. For example if the user chose yes with MB 0.5 then the var_val entry would be [[yes [0.5 0.5]] [no [-0.5 -0.5]]]. Should the user reply with Unknown then each of yes and no will assume certainties of 0.

To access the value of a variable two main functions are provided. The simplest function is the equality operator <=> which will test a variable against a test value and will return the CF range attached to that value. For example the test (variable <=> "yes") on the above variable would return [0.5 0.5] as its CF range. A further access function is provided for use in conditional statements in demons. This function will return a boolean value which is necessary to fire a demon. The function is called 'predicate_checks' and has three arguments, a predicate for the test, a variable to be tested, and the value being tested. The syntax is as follows:

```
predicate_checks("pred",var_name,"value");
```

The pre-defined predicate ranges are shown in Figure 5.7. It is possible to add to these by consing new values into the list 'predicates_list'. each entry must have the structure [pred_name [<lower val bounds> <upper val bounds>]]. Where <lower val bounds> represents two numbers between which the lower CF of the test value must lie, and similar for <upper val bounds>. For example, the 'likely' predicate might have an entry [likely [0.2 1.001 0.2 1.001]]. Notice that the testing function assumes that the value is tested as:

lower limit <= lower CF < upper limit

New predicates would be added in the following syntax:

[newpred1 [0.1 0.1001 0.1 0.1001]] :: predicates_list -> predicates_list;

[newpred2 [0.2 0.2001 0.2 0.2001]] :: predicates_list -> predicates_list;

Menu Questions

Questions of type 'menu' are only slightly different to those of type 'simple'. The main difference being that the user is given more than two possible responses and must in fact choose from a list of exclusive possibilities. Only two entries in the definition are diferent. The main change is the addition of a MENU entry which has the following syntax:

```
MENU [['pos 1' internal_1]
      ['pos 2' internal_2]]
```

This entry can be added at any point in the definition and the interpreter will sort it out. The 'pos 1' and 'pos 2' elements are the strings which will be printed for the user whilst the internal_1 and internal_2 elements are the internal values which the system will use. This allows for more expressive menu entries which should be of assistance to the user. The only other change to the definition is replacement of the EXPECTS entry with some suitable value such as ['choose a number, or (U)nknown.'] to be used as a prompt.

Once the user chooses a menu item the system will ask for a MB value. This will give the CF range for the chosen value. The remaining certainty will then be assigned to the other menu entries. For example in a menu with entries [red green blue] the user might choose red with a CF of 0.7. The resulting var_val entry would be [[red [0.7 0.7]] [green [-1 -0.7]] [blue [-1 -0.7]]]. Note that the other values assume a spread range of CF since there is no way of anticipating which of the entries is causing the user's uncertainty, if any.

List Type Questions

The final question type is the list question. This construct is used in order to collect data which will naturally form a list of related pieces of information. For example, in METEX the system requires to discover what minerals, and in what quantity, are contained in the rock. The resultant list of minerals forms a record of the modal composition of the rock.

Questions of type 'list' are defined in a similar way to those of type 'simple' described earlier. Apart from altering the TYPE entry, only the EXPECT entry must be altered by the knowledge engineer. The required alteration will define, for the system, the required structure of the data list and also the constraints upon each entry in the list. The syntax of the new EXPECT entry is as follows:

['a string forming the prompt' [a list to allow the checking of the input]]

For example, in METEX the mode is requested and the prompt indicates the required structure for each mineral entry:

['<mineral name> <quantity> <return>' [?y:check_abrev ?x:isinteger]]

The first entry of the list is printed to prompt the user as before. The second entry is used by the POP11 pattern matcher to check the structure and content of the entered data. The user will be required to enter each mineral and its quantity on a separate line so that

each entry is checked on input before being assigned to the storage variable. The user is allowed to enter mineral names as any standard abbreviation or in full. The system then checks this against a list of abbreviations to ensure that the internally stored mineral name is the correct full name. For example had the user typed 'plag 50' and 'qtz 50' then the system would build a mode list with the correct mineral names, e.g. [[plagioclase 50] [quartz 50]]. Readers are again referred to the POPLOG help system, or the available POP11 text books (Barrett et al. 1985, Ramsey & Barrett 1987, Burton & Shadbolt 1987, Leventhol 1987), for a full description of the pattern matcher and its features as used here.

Data Directed Questioning

As was indicated, in the earlier description of 'simple' questions, it is possible to define demons within question definitions. This feature allows the knowledge engineer to have the system follow particular lines of questioning if the data for a particular specimen indicates that such a line should be followed. For example, in METEX the user is asked if the rock is 'foliated' and then is asked what type of foliation is shown. There would be little point in attempting to ascertain the type of foliation in an unfoliated rock. As well as being pointless this would appear as completely unintelligent behaviour to the user. To this end it is possible to use demons to check the values of data in the system, either propositions or variables, and based upon the data, add new questions, or indeed new frames, to the system agenda.

A2.3 Compiling and Saving Knowledge Bases

Once a Knowledge Base has been built, either as a full knowledge base or as a subset of the whole, it must be loaded into the interpreter and tested. This testing will check firstly for errors made while typing in the knowledge base, and secondly for fundamental errors in the encoded knowledge. In order to carry out such testing the knowledge must first be compiled into the internal representation used by KRILL. The

first step in the compilation is to ensure that the interpreter is cleared of any previous knowledge. This is achieved by running the macro 'newdomain' which clears all global data entries and runs the POP11 garbage collector. The interpreter is now ready to load in a new knowledge base. This is achieved by running the macro 'lkb' (load knowledge base) which has as argument the name of the file to be compiled. This file should be the new knowledge base and should be located in the directory set aside for knowledge bases. The syntax for running the 'lkb' macro is as follows:

lkb <filename>

The 'lkb' macro assumes that files will end in the qualifier '.kb', for example 'fred.kb', so that it is not necessary to have this part of the name in the argument <filename>.

Once a knowledge base has been loaded successfully it can be saved as a compiled image which can be loaded much more quickly for future use. However, it is probably not worth working on a saved image until the knowledge base has been debugged to some satisfactory level. To save an image the macro 'saveimage' should be run. This macro requires a filename as argument and will save the image of the compiled knowledge base in a file called '<filename>.psv'. This file is then reloaded by using the macro 'getimage' again with the filename as argument, please note that again the '.psv' qualifier is automatically added. The sequence of commands is as follows:

: saveimage fred <CR>

** false

: getimage fred <CR>

** true

Please note that saved image files should not under any account be sent to the printer. The author can not comment on other systems but under VMS serious system

errors occur with the computer aborting all of its print queues. This will not be popular with the system managers! Any reader requiring fuller details on saved images should refer to the POPLOG help system, or the POP11 textbooks, concentrating specifically on the functions 'syssave' and 'sysrestore'.

A2.4 Debugging a Knowledge Base

When a knowledge base is loaded errors can occur, either during compilation, or later while running the knowledge base. It is proposed to give some guidelines as to how these errors can be detected and rectified by using some simple tools provided in the interpreter.

A2.4.1 Compilation Errors

When a new knowledge base, or *an updated* knowledge base, is first loaded it is likely that there will be some syntactic errors in the source file. The compilation routines can detect most errors, such as missing entries in definitions, but because of the nature of the loading routines some errors can not be detected. The most common error is the omission of a terminator on some expression or definition. This leads to the system attempting to load the remainder of the knowledge base as part of the unterminated structure. An example of the situation in which this type of error can occur is in the definition of TEXT in a PROPOSITION or QUESTION where a missing list bracket will lead to the whole knowledge base being loaded as part of the list. A similar situation will occur with BOOKKEEPING entries which are not correctly terminated with a semi-colon. These errors will only show up when the compilation takes an unusually long time as the system essentially goes into an endless wait time looking for input of the required terminator.

To find the cause of the error it is possible to read through the knowledge base looking for the fault or the interpreter can be used to narrow down the area in which the error has occurred. To do this the interpreter should be cleared using the 'newdomain' command. The debugging aid can then be activated by typing:

1 -> debug_level;

This sets the debug routine to report messages at various level between 0 = 'no messages'(default), and 3 = 'all messages'. Errors can be detected more easily by working through the levels one at a time gradually narrowing down the possible source. Now the knowledge base should be reloaded using 'lkb'. As each major definition is loaded then the beginning and end will be announced. If two major units appear to overlap then the error is in the first of the two. By increasing the level of the debug messages then it is possible to follow loading down to the level of individual propositions and rules. The output during loading should be as follows:

** [reading FRAME rock]

** [finished reading FRAME rock]

** [reading FRAME metamorphic]

** [finished reading FRAME metamorphic]

If the error were in the FRAME 'rock' then the output might be:

** [reading FRAME rock]

** [reading FRAME metamorphic]

** [finished reading FRAME metamorphic]

This process of loading continues until the knowledge base can be loaded without errors at which point the system can be run by typing the command 'krill'. It is advisable to reset the variable 'debug_level' to 0 before running the system.

A2.4.2 Run Time Errors

Run time errors fall into three basic categories:

1. Errors in knowledge structures.

2. Errors in POP11 code used in procedural attachments.

3. Errors in the Knowledge itself.

Since errors in the knowledge itself can only be resolved by the expert it is proposed to concentrate on errors of type 1 and 2 here.

Errors in Knowledge Structures

If the reference used to activate any major knowledge structure does not correspond exactly to the name given to the structure then the interpreter will not be able to follow the reference. If this occurs when the result of a slot should activate a subclassification, then rather than carry out the required task the next valid slot will be attempted or the consultation ended. However, at least in this situation some level of result is gained. If the slot had in fact incorrectly referenced the required ruleset, then the system would tend to fail completely. In order to follow the activation of knowledge structures an aid can be activated to use the top screen window as a report facility. This is achieved by setting the variable 'twsecure' to be false (default true). After this has been done then when the system is run the activation of each frame, slot and ruleset will be reported.

A further area in which errors can occur is in the assignment of goals to a ruleset. If a slot activates a ruleset and assigns a goal for which no proposition definition exists then

the system will fail. In this case the POP11 interpreter will complain that a number was found instead of a record of type ER_atom. This is fairly simple to remedy. Either define the goal proposition or remove the assignment of the goal. It should be noted that goal propositions should not be asked of the user, a situation which would in fact be pointless since a question definition would be of more use than a ruleset here. If however it is desired to ask about some goals as an interim measure during development, then rules should be written to force a new proposition other than the goal itself to be asked. This measure will prevent the interpreter from failing.

Code Errors in Procedural Attachments

As has been described previously, it is possible to add domain specific, POP11 code to the knowledge base. This can be done as procedural attachments either via demons or via expression clauses in proposition definitions. The runnable code is either in the form of a small clause within the procedural attachment, or a larger defined procedure which is called from the procedural attachment.

Code which appears in the body of a demon, or an expression, will only show up syntactic errors when an attempt is made to run the code. This is because the POP11 function 'popval' is used to run this code so that it is only compiled at run time. Errant code can however be edited as with normal POP11 code and the knowledge base reloaded and tested. Major procedure definitions will be compiled at the time of loading a knowledge base so that syntactic errors will show up immediately.

The author is unable to ensure that all normal functions will work through procedural attachments. This is because the 'popval' function reassigns the variable 'proglist' to be a closed list containing the code from the procedural attachment. Normally 'proglist' is a dynamic list attached to the standard input stream, the terminal during runtime. The attachment of 'proglist' to the terminal allows normal input functions to be used, i.e. functions which read proglist for data. However, once 'proglist' has been

assigned a closed list, some of the input routines no longer operate correctly. Currently problems have only been experienced with the 'readline' function and a new version of this has been provided, 'readline1', which as with 'readline' returns a list from the terminal. This list can be used to contain POP11 code which can be compiled at the required time via 'popval'. Similar problems will occur with any functions which require input to be compiled, for example 'listread'. If such functions are required in POP11 code called via 'popval' then it will be necessary to write a new version of the required function in terms of 'readitem' and 'readline1'. For example, a single line string could be read into a variable using the code:

```
popval(readline1) -> variable;
```

Readers who have difficulties with code in this situation are referred to the POPLOG help system, and the POP11 texts, concentrating on details of compilation in POP11 and the function 'popval'.

APPENDIX 3

Knowledge Base Syntax Summary

This Appendix is intended as a reference guide to potential knowledge engineers. In the first section of the Appendix all of the major variables and procedures of use in building a knowledge base are briefly described, indicating their function and usage. The second section of the Appendix summarizes the required syntax of the main knowledge structures available.

A3.1 Variables and Functions

A3.1.1 Variables

debug_level	-Integer(default=0). Values between 0 and 3 indicate increasing levels of debugging information. 0 ='no messages', 3='all messages'.
KB_COPYR	-POP11 string, length <28 characters. Copyright identity for the Knowledge Base.
KB_ID	-POP11 string, length <40 characters. Printed in topmost screen window as knowledge base identifier.
KB_vers	-POP11 string, length ~5 version number of the knowledge base.
predicates_list	-POP11 list. A list of predicates and their associated CF range limits. Structure of individual entries: [predicate [Ll Lu Ul Uu]] where a value matches the predicate if its lower CF limit satisfies Ll<=Lower CF<Lu and the upper CF limit satisfies Ul<=Upper CF<Uu.

sem_new_facts	-POP11 list. A list of the nodes to be used for propagation when a semantic net is activated.
top_level_frame	-POP11 word. The frame which is to be used as the begin point in the search tree.
twsecure	-Boolean(default=true). If set to false allows the upper screen window to be used for reporting the activation of frames, slots and rulesets.

A3.1.2 Functions

@	-Usage (x @ y). Tests a numerical value x against a fuzzy set description y. Returns a CF value according to the order of membership of x in y.
<=>	-Usage (x <=> y). Tests a domain variable 'x' against a value 'y'(a POP11 word) returning the CF with which the variable 'x' is taken to have the value 'y'.
add_agenda	-Usage (add_agenda(x)). Adds a knowledge structure identifier 'x'(a POP11 word) to the active control agenda.
getimage	-Usage (getimage <filename>). Restores a saved image to memory. Image retrieved from file '<filename>.psv'.
krill	-Usage (krill). Runs the currently loaded knowledge base.
lkb	-Usage (lkb <filename>). Loads a knowledge base stored in the file '<filename>.kb'.
newdomain	-Usage (newdomain). Clears the current knowledge base from memory.

predicate_checks -Usage (predicate_checks(x,y,z)). Tests an item 'y' against a predicate 'x'(a POP11 word). If 'y' is a proposition name then its CF is tested against the predicate. If 'y' is a domain variable then the CF of value 'z'(a POP11 word) is tested against the predicate. Returns <true> or <false>.

saveimage -Usage (saveimage <filename>). Saves a compiled image in the file '<filename>.psv'.

A3.1.3 Special Variable Types

Domain Variables -Used to store data from pre-defined questions.

Structure:

var_val	var_used	var_found	var_stat
list1	list2	word1	word2

Where:

list1 -Contains the possible values of the variable along with a CF range.structure of this list is:

1. [[yes [1 1]] [no [0 0]]] for simple questions.
2. [[red [1 1]] [blue [0 0]] [green [0 0]]] for menu questions.
3. [[[[quartz 20] [biotite 30] [muscovite 50]] [1 1]]] for results of list questions.

word2 -Stores the status of the variable. Either "UNASKED", "ACTIVE" or "SUSPENDED". Useful when testing a variable status in order to fire a demon.

list2, word1 -Both used by the interpreter but of no value to the Knowledge Engineer.

Fuzzy set descriptors -Used to store the set membership of particular values to some fuzzy term for use by the '@' operator. This term is domain dependant.

Structure:

[[[Lval1 Uval1] CF1]

:

[[Lvaln Uvaln] CFn]]

Where:

Lvaln -Lower limit of values which will assume the indicated CF of membership of the overall set.

Uvaln -Lower limit of values which will assume the indicated CF of membership of the overall set.

CFn -The CF which will be returned as the certainty that some value falls in the set of values which describes the fuzzy term.

This list is assigned to a variable which has as its name the fuzzy term.

A3.2 Knowledge Base Syntax

Comments -Normal POP11 comments within the symbols /* and */

<frame> :=

FRAME <id>

VIA <var_name 1> ... <var_name n>;

INTRO <string 1> ... <string n>;

SLOTS <slot 1> ... <slot n> ENDSLOTS

DEMONS <demon 1> ... <demon n> ENDDEMONS

ENDFRAME

<slot n> :=

SLOT <id>

ASSIGNS <var_name>

INTRO <string 1> ... <string n>;

PREACTS [<preact 1> ... <preact n>]

FROM <ruleset id>

CAN_BE [<goal 1> ... <goal n>]

DEFAULT <default result>

DEMONS <demon 1> ... <demon n> ENDDEMONS

ENDSLOT

<preact n> := <frame>

:= <question>

<question> :=

QUESTION <id>

TYPE <q_type>

MENU <q_menu>

TEXT [<string 1> ... <string n>]

EXPECT [<string 1> <constraint>]

ASSIGNS <var_name>

HELP <string 1> ... <string n>;

B_K <BK_string>;

DEMONS <demon 1> ... <demon n> ENDDEMONS

ENDQUESTION

<demon n> :=

D_ <id> WHEN [<boolean expression>]

DO [<expression>];

<ruleset id> := POP11 word. Pointer to a <ruleset>.

<ruleset> :=

E_RULES <id>

CONCERNS <var_name 1> ... <var_name n>;

B_K <BK_string>;

PROPOSITIONS <prop 1> ... <prop n> ENDPROPS

ENDE_RULES

<prop n> :=

PROP <prop name>;

TEXT <list 1> ... <list n>;

RULES <rule 1> ... <rule n> ENDRULES

EXP <expression>;

ASKABLE

NOT_ASKABLE

ENDPROP

<rule n> :=

ER_ <rule number> <BK_string>;

IF <premise clause>

THEN <wight clause> <prop name>;

<premise clause> := <premise> <comb> <premise>

:= <prop name>

<premise> := <premise clause>

:= <|* premise clause *|>

:= <prop name>

<|* premise clause *|> := a clause which has been isolated in parenthesis.

<comb> := AND

:= AND NOT

:= OR

:= OR NOT

<weight clause> := <wt>

:= [<wt> <wt>]

<wt> := real number between -1 and 1.

<prop name> := string of POP11 words giving a symbolic proposition. Not necessarily a POP11 string.

<pattern block> :=

RULE_PATTERNS

<pattern 1> ... <pattern n>

END_RULE_PATTERNS

<pattern n> :=

PATTERN <template>

TEXT <list 1> ...<list n>;

EXP [<expression>];

RULES <rule 1> ... <rule n> ENDRULES

ENDPATTERN

<template> := POP11 matcher template to instantiate variables used in expressions an text lists as well as in rules.

<semantic> :=

SEMANTIC <id>

<semantic link 1> ... <semantic link n>

ENDSEMANTIC

<semantic link n> := SR_ <node> ==-> <node>;

<node> := POP11 word. a node in the semantic net.

<string n> := a POP11 string up to 80 characters in length.

<BK_string> := a 'string' (not POP11) of words giving information on source of knowledge.

<id> := POP11 word. Identification of a major knowledge structure.

<goal n> := POP11 word giving a system goal.

<preact n> := identifiers of Knowledge items to be treated as preactions to a slot.

<q_type> := simple

:= menu

:= list

<q_menu> := [<menu entry 1> ... <menu entry n>]

<menu entry n> := [<string n> <internal>]

<internal> := POP11 word. Internal value for a variable.

<expression> := a piece of valid POP11 procedural code.

<bool expression> := a piece of valid POP11 code which produces a a result either <true> or <false>.

<list n> := a POP11 list of text.

<rule number> := a real number identifier for a rule.

REFERENCES

Adams, J.B. (1984), "Probabilistic Reasoning and Certainty Factors" in Buchanan, B.G. & Shortliffe, E.H. (eds), *Rule Based Systems - The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, 263-271.

Aikins, J.S. (1980), Prototypes and Production Rules: a Knowledge Representation for Computer Consultations. Ph.D. dissertation, Department of Computer Science, Stanford University.

Aikins, J.S. (1984), "A Representation Scheme using Both Frames and Rules", in Buchanan, B.G. & Shortliffe, E.H. (eds), *Rule Based Systems - The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, 424-440.

Alexander, I.F. (1985), "An Expert System for Palaeontology", *Tertiary Research* 7(1), 1-11.

Alty, J.L. & Coombs, M.J. (1984), *Expert Systems - Concepts and Examples*, NCC Publications.

Anderson, J., Bandler, W., Kohout, L.J. & Trayner, C. (1985), "The Design of a Fuzzy Medical Expert System", in Gupta, M.M, Kandel, A., Bandler, W. & Kiszka, J.B. (eds), *Approximate Reasoning in Expert Systems*, Elsevier North-Holland, 689-704.

Applebaum, L. & Ruspini, E.H. (1985), "Aries: An Approximate Reasoning Inference Engine", in Gupta, M.M, Kandel, A., Bandler, W. & Kiszka, J.B. (eds), *Approximate Reasoning in Expert Systems*, Elsevier North-Holland, 745-765.

Baker, D.J. (1985), "Dipmeter Advisor - An Expert Log Analysis System at Schlumberger", in Winston, P.H. & Prendergast, K.A. (eds), *The AI Business*, MIT Press, 51-66.

Barrett, R., Ramsay, A. & Sloman, A. (1985), *POP-11 a Practical Language for Artificial Intelligence*, Chichester: Ellis Horwood.

Batnagar, R.K. & Kanal, L.N. (1986), "Handling Uncertain Information: A review of Numeric and Non-numeric Methods", in Kanal, L.N. & Lemmer, J.F. (eds), *Uncertainty in Artificial Intelligence*, Elsevier North-Holland, 3-26.

Ben-Bassat, M. (1985), "Expert Systems for Clinical Diagnosis", in Gupta, M.M, Kandel, A., Bandler, W. & Kiszka, J.B. (eds), *Approximate Reasoning in Expert Systems*, Elsevier North-Holland, 671-688.

Best, M.G. (1982), *Igneous and Metamorphic Petrology*, W H Freeman and Co.

Bonnet, A. & Dahan, C. (1983), "Oil Well Data Interpretation Using Expert Systems and Pattern Recognition Techniques", *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, Karlsruhe-West Germany, 185-189.

Buchanan, B.G. & Shortliffe, E.H. (eds) (1984), *Rule Based Systems - The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley.

Bundy, A. (1980), *Artificial Intelligence an Introductory Course* (revised Edition), Edinburgh University Press.

Burton, M. & Shadbolt, N. (1987), *POP-11 Programming for Artificial Intelligence*, Wokingham: Addison Wesley.

Campbell, A.N., Hollister, V.F., Duda, R.O. & Hart, P.E. (1982), "Recognition of a Hidden Mineral Deposit by an Artificial Intelligence Program", *Science* **217**, 927-929.

Clancey, W.J. (1979), "Dialogue Management for Rule-based Tutorials", *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, Tokyo, 155-161.

Clancey, W.J. & Lestinger, R. (1981), "NEOMYCIN: Reconfiguring a Rule-based Expert System for Application to Teaching",*Proceedings of the 7th International Joint Conference on Artificial Intelligence*, Vancouver, 829-836.

Clancey, W.J. (1983), "The Epistemology of a Rule-Based Expert System - a Framework for Explanation.", *Artificial Intelligence* **20**, 215-251.

Davis, R. & Lenat, D.B. (1982), *Knowledge-Based Systems in Artificial Intelligence*, McGraw-Hill.

Davis, R. (1984), "Interactive Transfer of Expertise", in Buchanan, B.G. & Shortliffe, E.H. (eds), *Rule Based Systems - The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, 171-205.

Davis, R. & Buchanan, B.G. (1977), "Meta-level Knowledge: Overview and Applications", *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Cambridge-Massachussets, 920-927.

Davis, R., Buchanan, B.G. & Shortliffe, E.H. (1977), "Production Rules as a Representation in a Knowledge-Based Consultation System", *Artificial Intelligence* **18**, 15-45.

Davis, R. & King, J. (1977), "An Overview of Production Systems", in Elcock, E.W. & Michie, D. (eds), *Machine Intelligence* **8**, 300-332.

Davis, R., Austin, H., Carlbom, I., Frawley, B., Pruchnik, P., Sneiderman, R. & Gilreath, A. (1981), "The DIPMETER ADVISOR: Interpretation of Geological Signals", *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, Vancouver, 846-849.

Davis, R. (1987), "Knowledge-Based Systems: The View in 1986", in Grimson, W.E.L. & Patil, R.S. (eds), *AI in the 1980s and Beyond*, MIT Press, 13-42.

Denham, L.R. (1985), "Expert Systems in Seismic Exploration", *Proceedings of the 17th Annual Offshore Technology Conference*, Houston Texas, 203-207.

Dubois, D. & Prade, H. (1985), "Fuzzy Cardinality and the Modeling of Imprecise Quantification", *Fuzzy Sets and Systems* **16**, 199-230.

Duda, R.O., Hart, P.E., Nilsson, N.J.& Sutherland, G.L.(1978), "Semantic Network Representations in Rule-Based Inference Systems", in Waterman, D.A. & Hayes-Roth, F.(eds), *Pattern-Directed Inference Systems*, Academic Press, 203-221.

Duda, R., Gaschnig, J. & Hart, P. (1979), "Model Design in the Prospector Consultation System for Mineral Exploration", in Michie, D. (ed), *Expert Systems in the Micro Electronic Age*, Edinburgh University Press, 153-167.

Erman, L.D., Hayes-Roth, F., Lesser, V.R. & Reddy, D.R. (1980), "The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty", *Computing Surveys* **12**, 213-253

Fiegenbaum, E.A., Buchanan, B.G. & Lederberg, J. (1971), "On Generality and Problem Solving: A Case Study using the Dendral Program", in Meltzer, B. & Michie, D. (eds), *Machine Intelligence* **6**, American Elsevier, 165-190

Fiegenbuam, E.A. (1977), "The Art of Artificial Intelligence: 1 Themes and Case Studies of Knowledge Engineering", *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Cambridge-Massachussetts, 213-222.

Fiegenbaum, E.A. (1979), "Themes and Case Studies of Knowledge Engineering", in Michie, D. (ed), *Expert Systems in the Micro Electronic Age*, Edinburgh University Press, 3-25.

Fry, N. (1984), *The Field Description of Metamorphic Rocks*, Geological Society of London Handbook Series, Open University Press.

Gammerman, A. & Creaney, N. (1986), "Modelling Uncertainty in Expert Systems", *Proceedings of The 2nd International Expert Systems Conference*, London 1986, 265-274.

Garvey, T.D., Lawrance, J.D. & Fischler, M.A. (1981), "An Inference Technique for Integrating Knowledge from Disparate Sources", *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, Vancouver, 319-325.

Gashnig, J. (1982), "Application of the PROSPECTOR System to Geological Exploration Problems", in Hayes, J.E., Michie, D. & Pao, Y-H. (eds), *Machine Intelligence* **10**, 301-323.

Gershman, A. (1982), "Building a Geological Expert System for Dipmeter Interpretation", *Porceedings of the European Conference on Artificial Intelligence-82*, 139-140

Gillen, C. (1982), *Metamorphic Geology, An Introduction to Tectonic and Metamorphic Processes*, George, Allen and Unwin.

Gordon, J. & Shortliffe, E.H. (1984), "The Dempster-Shafer Theory of Evidence", in Buchanan, B.G. & Shortliffe, E.H. (eds), *Rule Based Systems - The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, 272-292.

Gordon, J. & Shortliffe, E.H. (1985), "A Method for Managing Reasoning in an Hierarchical Hypothesis Space", *Artificial Intelligence* **26**, 323-357.

Greensmith, J.T. (1978), *Petrology of the Sedimentary Rocks* (6th edition), George, Allen & Unwin.

Harmon, P. & King, D. (1984), *Expert Systems Artificial Intelligence in Business*, New York: John Wiley.

Hatch, F.H., Wells, A.K. & Wells, M.K. (1972), *Petrology of the Igneous Rocks* (13th edition), George, Allen & Unwin.

Hawkes, D.D. (1985), "INTAL- an Expert System for the Identification of Igneous Rocks in the Hand Specimen", *Geological Journal* **20**, 367-375.

Hayes-Roth, B. (1985), "Blackboard Architecture for Control", *Artificial Intelligence* **26**, 251-321.

Heckerman, D. (1986), "Probabilistic Interpretation for MYCIN's Certainty Factors", in Kanal, L.N. & Lemmer, J.F. (eds), *Uncertainty in Artificial Intelligence*, Elsevier North-Holland.

Hietanen, A. (1967), "On the Facies Series in Various Types of Metamorphism", *Journal of Geology* **75**, 187-214.

Hunt, E.B. (1975), *Artificial Intelligence*, London, New York: Academic Press.

Hyndman, D.W. (1985), *Petrology of Igneous and Metamorphic Rocks* (2nd edition), McGraw-Hill.

Jackson, K.C. (1970), *Textbook of Lithology*, New York, Maidenhead : McGraw-Hill.

Joplin, G.A. (1968), *A Petrography of Australian Metamorphic Rocks*, Melbourne: Angus & Robertson.

Kidd, A.L. (1985), "What do Users Ask? - Some Thoughts on Diagnostic Advice", in Merry, M. (ed), *Expert Systems 85, Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems*, Cambridge University Press, 9-19.

Kraft, A. (1985), "XCON: An Expert Configuration System at Digital Equipment Corporation", in Winston, P.H. & Prendergast, K.A. (eds), *The AI Business*, MIT Press, 41-50.

Leventhol, J. (1987), *Programming in POP-11*, Blackwell Scientific Publications.

Mason, R. (1978), *Petrology of the Metamorphic Rocks*, George, Allen & Unwin.

Mellish, C.S. (1985), "Generalised Alpha-Beta Pruning as a Guide to Expert System Question Selection", in Merry, M. (ed), *Expert Systems 85, Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems*, Cambridge University Press, 31-41.

McDermot, J. (1982a), "R1: A Rule-Based Configurer of Computer Systems", *Artificial Intelligence* **19**(1), 39-88.

Shortliffe, E.H. & Buchanan, B.G. (1975), "A Model of Inexact Reasoning in Medicine", *Mathematical Biosciences* 23, 351-379.

Shortliffe, E.H. (1976), *Computer Based Medical Consultations: MYCIN*, (Elsevier Computer Science Library, Artificial Intelligence series 2), Elsevier.

Shortliffe, E.H. (1984), "A model of Inexact Reasoning in Medicine", in Buchanan, B.G. & Shortliffe, E.H. (eds), *Rule Based Systems - The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, 233-262.

Smith, R.G. & Baker, J.D. (1983), "The Dipmeter Advisor System - A case Study in Comercial Expert Systems Development", *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, Karlsruhe-West Germany, 122-129.

Smith, D.E & Clayton, J.E. (1984), "Another Look at Frames", in Buchanan, B.G. & Shortliffe, E.H. (eds), *Rule Based Systems - The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, 441-452.

Spry, A. (1969), *Metamorphic Textures*, Oxford: Pergamon.

Suk, M. (1983), *Petrology of Metamorphic Rocks*, Developments in Petrology 9, Elsevier Scientific.

Szolovits, P. & Pauker, S. (1978), "Categorical and Probabilistic Reasoning in Medical Diagnosis", *Artificial Intelligence Journal* 11(1,2), 115-154.

Szolovits, P. (1987), "Expert Systems Tools and Techniques: Past, Present and Future", in Grimson, W.E.L. & Patil, R.S. (eds), *AI in the 1980s and Beyond*, MIT Press, 43-74.

Trigoboff, M. (1976), "Propogation of Information in a Semantic Net", *Proceedings of Artificial Intelligence and the Simulation of Behaviour conference (AISB)*, Edinburgh.

Trigoboff, M. & Kulikowski, C.A. (1977), "IRIS: A System for the Propogation of Inferences in a Semantic Net", *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Cambridge-Massachusetts, 274-280.

Turner, F.J. (1968), *Metamorphic Petrology, Mineralogical and Field Aspects* (1st edition), McGraw-Hill.

Turner, F.J. (1981), *Metamorphic Petrology, Mineralogy, Field and Tectonic Aspects* (2nd edition), Washington, London: Hemisphere.

van Melle, W. (1979), "A Domain Independant Production Rule System for Consultation Programs", *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, Tokyo, 923-925.

van Melle, W., Shortliffe, E.H. & Buchanan, B. (1984), "EMYCIN: A Knowledge Engineer's Tool for Constructing Rule-Based Expert Systems", in Buchanan, B.G. & Shortliffe, E.H. (eds), *Rule Based Systems - The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, 302-313.

West, J. (1985), "Toward an Expert System for Identification of Minerals in Thin Section", *Journal of the International Association for Mathematical Geology* **17**(1), 743-753.

Winston, P.H. (1984), *Artificial Intelligence* (2nd edition), Adison-Wesley.

Winkler, H.G.F. (1979), *Petrogenesis of Metamorphic Rocks* (5th edition), New York: Springer.

Yan, J. (1986), "Identifying Depositional Environment Structure: An Expert System Approach Using Object Oriented Programming and Model Driven Verification", *Proceedings of the 2nd International Expert Systems Conference*, London, 441-449.

Zadeh, L.A. (1978), "Fuzzy Sets as a basis for a Theory of Possibility", *Fuzzy Sets and Systems* **1**, 3-28.

Zadeh, L.A. (1985), "The Role of Fuzzy Logic in the Management of Uncertainty", in Gupta, M.M, Kandel, A., Bandler, W. & Kiszka, J.B. (eds), *Approximate Reasoning in Expert Systems*, Elsevier North-Holland, 3-31.