THE APPLICATION OF EXPERT SYSTEMS TO SMALL SCALE MAP DESIGN

BY

DAVID FORREST

A thesis submitted for the degree of Doctor of Philosophy
University of Glasgow
Department of Geography & Topographic Science
June 1995

ABSTRACT

The increased availability of inexpensive computer mapping programs in recent years has lead to a great increase in the number of map authors and the number of maps being produced, but does not however appear to have lead to more widespread knowledge of cartographic design theory. The large number of poorly designed maps created by users of these computer systems indicates that there is a lack of knowledge of how to design maps. These poorly designed maps are not the fault of the computer programs, since most programs do have the capability of producing well designed maps when used by someone knowledgeable in map design. Rather, the problem lies with map authors who are not skilled in cartographic design and who would probably never produce a map by conventional means; but would contract a cartographer to produce it. What is required are programs to be used by naive map authors that are better able to produce reasonably well designed maps, or at least maps which do not break the most fundamental rules of map design. The area of computer science devoted to producing programs that include knowledge of how an expert solves a problem is that of Expert Systems. An Expert System is essentially a program which includes a codified form of the rules that an expert uses to solve a problem. Thus a cartographic design expert system would include the rules a cartographer uses when designing a map.

This study examines the fields of artificial intelligence and expert system to assess how they may best be applied to the map design problem. A comprehensive review of the application of expert systems in design, mapping generally and map design in particular is also provided. In order to develop an expert system, the problem or 'domain' must be defined in a relatively formal manner. A structure for describing geographic information and cartographic representation is developed and a model of the cartographic design process for application in expert systems is also described. Based on the models developed, a functional specification for a cartographic design expert system for small scale maps is produced, with the rules required for each stage in the design process being set out. The development of an expert system, written in Prolog, incorporating these rules is then described in some detail. Details of how the Prolog language can be applied to a specific problem, colouring the political map, are also given.

It has been found that as long as realistic goals are set and that the system is limited either in scale or range of topics, it is possible to develop an operational cartographic design expert system. However, it must be recognised that a considerable amount of further development will be needed to bring such a system to market with the support structures and robustness that this entails.

ACKNOWLEDGEMENTS

As supervisor, John Keates was very influential in the early stages of this project. In particular his feedback on the models of cartographic representation and cartographic design developed in Chapter Five helped turn rather loose descriptions into rigorous, well defined statements. Advice was always available when sought, as was stimulating conversation about more general cartographic matters. Towards the later stages of writing up the project Alastair Morrison provided many useful comments and I am indebted to him for volunteering to become involved at a late stage. Yvonne Wilson produced figures 2.5, 3.1, 3.2, 3.3 & 3.5, all other figures are by the author.

This whole project would not have started without the co-operation and encouragement of Dr. Michael Staveley and Dr. Christopher Sharp, Dean and Head of Geography respectively, at Memorial University of Newfoundland. Their willingness to allow leave of absence, both paid and unpaid, was clearly more in my interest than theirs and I am grateful to them for this.

Funding for part of the early stages of this study was provided by a post graduate scholarship from the Natural Science and Engineering Research Council of Canada (NSERC).

A very important influence on my life has been my parents whose continued support, both moral and financial, has been of great value. What they have freely given over the years is much more than any child could reasonably expect. I hope I can provide as well for my children as they have for theirs.

To my children, Lillian and Euan, both born since the inception of this project, I hope that 'Daddy's working' will become a less frequent rebuke when they want to sit on my knee and play a game on the computer. And finally, to Pauline, who has spent many evenings alone while I battled with some aspect or other of developing the program and has often felt she played second fiddle to that ∅Λ∩ϒ! PhD, that eternity is at last over and perhaps we can again spend more time enjoying life together.

Table of contents

CHAPTER THREE

CHAPTER FOUR

CHAPTER FIVE

CHAPTER EIGHT

List of Figures

List of Tables

INTRODUCTION

In the last twenty years or so a great number of programs have been created to produce maps using a computer.[1] Most of the commonly available programs are for producing small scale statistical maps, but more recently there has been a significantly increased interest in using Geographic Information Systems for producing a wider variety of maps at a broader range of scales.[2] The continual decrease in hardware prices, particularly in association with the increased power of micro-computers now available, has brought the possibility of computer mapping to a much wider range of users.

The increase in the availability of computer mapping facilities has lead to a great increase in the number map of authors and the number of maps being produced, but does not however appear to have lead to more widespread knowledge of cartographic design theory. The large number of poorly designed maps created by map authors[3] using computer systems to produce their own maps indicates that there is a lack of knowledge of how to design maps. These poorly designed maps are not the fault of the computer programs, since most programs do have the capability of producing well designed maps when used by someone knowledgeable in map design. Rather, the problem lies with authors who are not

[1] Numerous terms have been used to describe maps produced with the aid of computers: computer cartography, computer mapping, computer assisted cartography, digital mapping, automation, etc. In the current study we are examining the design and production of maps for display on computer monitors (VDUs) or output on small-format plotters and printer/plotters. The term Computer Aided Cartography (cf. Computer Aided Design) will be used to refer to computers being used for the design and display of maps, whereas Computer Mapping will be used to refer to the broader use of computers in map making.

[2] A Geographic Information System (GIS) should include a database capable of holding geographic information, tools for analysing this data and the capability of mapping the results. Computer Aided Cartography could usefully be applied to the third part of such a system.

[3] The **Map Author** is one who conceives the map and often prepares the special topic data. He may then proceed to carry out the design and production, or pass this on to the **Cartographer**. The **System User** is the user of a Computer Mapping system. He may be the Map Author and/or the Cartographer. The **Map User** may be different from the Map Author and System User. The knowledge and experience of the intended Map User(s) will influence the map's design and production.

skilled in cartographic design and who would probably never produce a map by conventional means, but would contract a cartographer to produce it.

It is unlikely that the general level of cartographic education of most computer map authors will be greatly increased, therefore cartographers must strive to make the programs used by naive map authors better able to produce reasonably well designed maps, or at least maps which do not break the most fundamental rules of map design.

The area of computer science devoted to producing programs that include knowledge of how an expert solves a problem is that of Expert Systems. An Expert System is essentially a program which includes a codified form of the rules that an expert uses to solve a problem. Thus a cartographic design expert system would include the rules a cartographer uses when designing a map.

A long term goal would be to have a cartographic design expert system that could design any map at any scale, but current literature on expert systems suggests that at this time practical expert systems should be limited to narrow domains, i.e. the problem area must be defined within quite narrow margins. Several cartographic expert systems are currently under development. These have tended to concentrate on elements of the map or map design, e.g. name placement, line generalisation, solution of spatial conflicts, etc., and while these problems will all have to be solved in any realistic production system, there is a pressing need for the application of expert system techniques to more general design issues such as data selection, choosing an appropriate method of portraying a data set in map context, and generally trying to prevent the author from making poor design decisions when making the map.

Despite the huge amount of investment that has been made in Geographic Information Systems and Computer Mapping systems in recent years Buttenfield & Mackaness note that:

> ... the role of cartography in these systems has largely been ignored. Instead, the graphics packages and pen plotters have 'replaced' cartographers and their scribing tools. Failure to accommodate sound principles of design into graphical defaults has resulted in the production of some appalling maps, examples of which abound in the literature.
>
> (1991; 439)

It would seem, therefore, that a system that only went as for as having sensible defaults for mapping would be a significant step forward. But it is not sufficient to simply enter a series of standard unvarying defaults into a system. To be useful, the defaults must change with the given circumstances. For example, many systems use a default of five classes for choropleth maps. When this is accepted, a map is produced with five sensible shadings (although not by all systems). If the number of classes is reduced to three, the first three shadings are used. This is less than optimal, but not too significant a problem. The situation is much worse when the number of classes is increased. If six classes are requested the resulting map uses the five standard symbols for the first five classes, but leaves the sixth class blank or assigns it something unsuitable. Clearly this is not satisfactory. Thus there is an urgent need to apply expert systems techniques to providing sensible and variable defaults to allow non cartographers to produce sensible maps.

Extending the use of expert systems techniques in mapping systems further than providing sensible defaults should allow the map author to concentrate on his primary objective, the making of a map to show some particular Information. He should be relieved of the cartographic problems which are of no direct concern or interest to him. The work reported on here illustrates how such a system could be developed. The general area considered here is the production of small scale maps as might be found in regional or educational atlases, or used to give national or regional overviews of a variety of topics. The maps are relatively small scale, but cover a wide range of subjects and representation methods.

First, Chapter One gives an overview of artificial intelligence research with the emphasis being given to expert systems. Chapter Two further explores the nature of expert systems, the fields in which they have been applied and their intended uses. This chapter also briefly discusses the man-machine interface, an important aspect of any interactive computer system.

A comprehensive review of the application of expert systems in design, mapping generally and map design in particular is provided in Chapter Three. The specific problem tackled by the system under development is defined in Chapter Four along with the sources of knowledge and the intended users and uses of the system.

In order to develop an expert system, the problem or 'domain' must be defined in a relatively formal manner. Chapter Five provides this background, developing a structure for describing geographic information and cartographic representation. A model of the cartographic design process for application in expert systems is also described. Following on from the descriptive information in Chapter Five, Chapter Six presents a functional specification of the system to be developed with the rules required for each stage in the design process being set out.

Chapter Seven is concerned with the background to the development of the actual system. The hardware and software environments are described and an introduction to the Prolog language is given. Chapter Eight describes the actual system in some detail, converting the rules presented in Chapter Six into a specific implementation, and concludes with an example run through the system, culminating with a map produced by it. Following on from this, further details of how Prolog can be applied to a specific problem, colouring the political map, are presented in Chapter Nine.

Finally, Chapter Ten summarises the findings of the study, discusses the problems that have been encountered and the limitations of the system as developed. Possible future developments are also discussed.

CHAPTER ONE

An Introduction to Artificial Intelligence and Expert Systems.

> Artificial intelligence is simply the transfer of intelligence to machines. Expert systems deal with a small area of expertise that can be converted from human to artificial intelligence.[1]

ARTIFICIAL INTELLIGENCE

Artificial Intelligence is a branch of computer science involved in studying mental faculties and reproducing them through the use of computational models. The use of the word 'intelligence' may in fact be misleading as the term tends to be used for mental feats of unusual creativity or cleverness, whereas most problems in Artificial Intelligence (AI) arise in attempting to recreate the mental capability of 'ordinary people'. AI is concerned with the general behaviour that goes along with intelligence; it is not limited to one particular method of producing 'intelligence', and the methods used may not be the same as people use (Charniak & McDermott, 1985; 7).

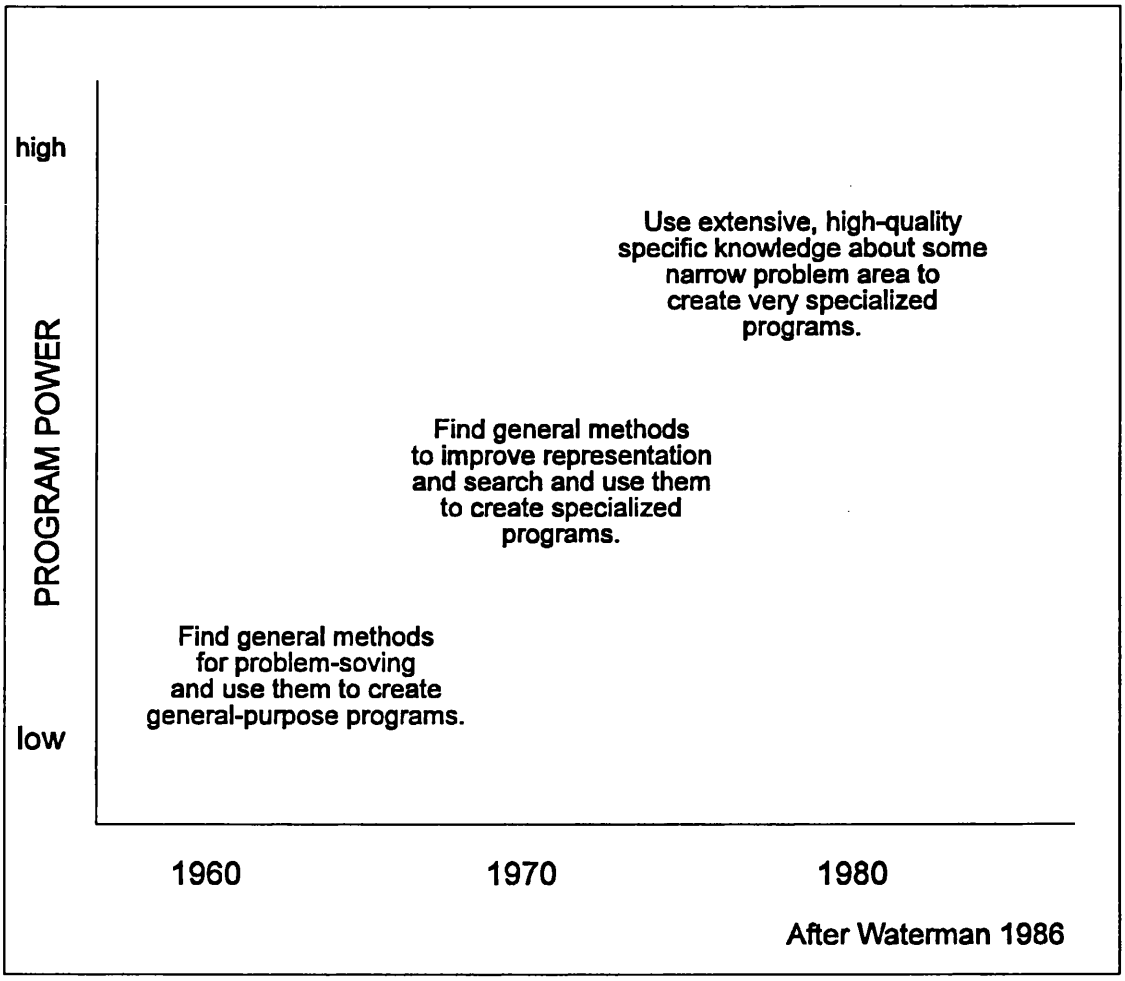
The ultimate goal of AI " ... is to produce human-like intelligence in a non-human machine" (James, 1984; 122). Whether or not this is achievable does not reduce the importance of developing programs that take us towards that goal. The divisions of AI research can be seen as the elements to be solved in producing such a machine. While there is no universal agreement on the subdivisions of AI, the major groupings are Expert Systems, Natural Language Processing, Pattern Recognition and Robotics. Other common sub-headings are Computer Vision and Machine Learning, although the former of these is frequently encompassed by Pattern Recognition and the latter is really an essential component of any system which claims to have artificial intelligence.

Of these divisions only Robotics is not of concern to this study. While one can imagine at some time in the future the possibility of a robot replacing a cartographer at a drafting table, this does not currently merit serious consideration. Of the other divisions, Expert Systems represents the 'brain' of a system and is the major focus of attention here. Natural Language Processing is an important aspect of communicating with the user of a computer program and ultimately should be incorporated in any system calling itself 'intelligent.' Pattern Recognition, which is

[1] Levine, R.I., Drang, D.E., Edelson, B. **A Comprehensive Guide to AI and Expert Systems.** New York: McGraw-Hill Book Company, 1986, pp.1.

important in many areas of the mapping sciences - especially remote sensing - does have some application to cartographic design.

While the general aim of AI research has not changed over its thirty year history, that is to produce programs that can in some way 'think', there has been a shift in emphasis from trying to find general methods for solving a broad range of problems to that of solving very specific problems with very highly specialised programs such as are illustrated by Figure 1.1. This chapter will examine some of the basics of Artificial Intelligence and its subdivisions: Current concepts of Expert Systems will be considered in detail, specifically, what they are, what they can do, and how they differ from conventional programs.


high
PROGRAM POWER
low
Use extensive, high-quality specific knowledge about some narrow problem area to create very specialized programs.
Find general methods to improve representation and search and use them to create specialized programs.
Find general methods for problem-soving and use them to create general-purpose programs.
1960
1970
1980

After Waterman 1986

Figure 1.1
The shifting focus of AI research

Intelligence

The idea of intelligence is not concerned solely with what can be done, but also how it is done (i.e. the style or manner). For example:

> 1. When confronted with messy, ill-defined problems and situations, and incomplete or uncertain information; an intelligent system should degrade gracefully as the degree of difficulty/ complexity/noise/incompleteness etc. increases, rather than merely 'crashing', or rejecting the problem. Degrading gracefully may involve being slower, less reliable, less general, less accurate, or producing less precise or complete descriptions etc.
>
> 2. Using insight and understanding rather than brute force or blind and mechanical execution of rules, to solve problems, achieve goals, etc.
>
> 3. Plans should not be created simply by applying pre-defined rules for combining primitive actions to achieve some goal, but should rely on the ability to use inference to answer hypothetical questions about 'what would happen if ..'. This should also play a role in the ability to make predictions, or test generalisations.
>
> 4. Conflicting goals should not be dealt with simply by means of a pre-assigned set of priority measures, but for example by analyzing the reasons for the conflict and making inferences about the consequences of alternative choices or compromises.
>
> (Sloman, 1984; 3)

There is a very thin division between programs that are clever and those that show artificial intelligence. Indeed, "... it is possible that there is no such thing as an intelligent program - just clever programs that become increasingly clever" (James, 1984; 116). It has been shown that by applying some simple rules one can give an impression of intelligence that would convince an innocent onlooker.

> This willingness [of people] to believe in the intelligence of computers has two important aspects. Firstly, ... it means that we can achieve some useful results without too much effort by borrowing some of the user's intelligence. Secondly, it cautions us that we must ourselves beware of becoming believers too easily.
>
> (James, 1984; 116)

Problem Solving

All branches of AI rely upon problem solving, to which there are two elements: Representation and Search. All of the approaches to problem solving require some sort of search for a solution. Conducting these searches as efficiently as possible is one of the aims of AI. However, before a search process can begin, the problem must be 'set up', or, in other words, a representation of the problem must be formulated.

> Usually one applauds a human problem solver not for conducting a fast and orderly search through all solution possibilities, but for looking at the problem in such a clever way that the solution seems elegantly simple.
>
> (Nilsson, 1971; 8)

There will often be alternative representations for the same problem, but unfortunately AI research is still directed at producing a generalised automatic method for the skilful formulation of problem representation.

Representation. The 'language' produced or operated upon during problem solving is known as the Internal Representation (Charniak and McDermott, 1985). This representation is, to some extent at least, an abstraction. The same representation may be embodied in a variety of different data structures, to make different operations efficient. It is normally assumed that it is easy to translate from one internal representation to another, and certainly easier than translation to and from external representations (i.e. questions and answers in English).

> The internal representation is used by an AI program in the following way:
>
> -When a program gets a statement, it translates it into an internal representation and stores it away.
>
> -When it gets a question, it translates it into an internal representation as well.
>
> -It uses the internal representation of the question to fetch statements from its memory.
>
> -It translates the answer back into English.
>
> (Charniak & McDermott, 1985; 11)

While this may seem more complex than simply storing the English, it is in fact more how people do things, in that we tend to remember the 'gist' of what we are told, long after we have forgotten the exact words. Specific knowledge representation methods for expert systems are discussed in Chapter 2.

Search. AI programs work by searching the internal representation of knowledge for a solution, often referred to as a goal or 'the goal state'. In human intelligence we can see the parallel to this as being a specific response to solve a particular problem. Our reactions to certain situations may appear to be automatic, but are the result of all our thought processes being directed to achieve a certain 'goal' (Levine et al., 1986; 4). We don't do things because we think, we think because we have things to do. This must always be considered when designing AI systems.

Typically the internal representation of a problem can be expressed as a tree structure or graph. This graph represents a structured series of nodes, each with an associated state descriptor. A solution is obtained by applying operators to these state descriptions until the 'goal state' is obtained (Nilsson, 1971). In the graph theory search process we have a start node which is associated with the initial state description. The successors of a node are 'calculated' using the operators that are applicable to the state description associated with the node, i.e. what process can be applied to the current situation to move towards the goal. For example, if our goal is to choose the most appropriate map projection we may first determine the purpose of the map. This will allow the appropriate special property (conformality, equivalence, etc.) to be selected.

The successor nodes of the current node are checked to see if they are the goal node (i.e. the associated state descriptor is the goal state or the solution required). If a goal node is not yet found the successor nodes are expanded to the next level and the process repeated until a goal is found. In our map projection example, this may involve determining the latitude of the area of interest in order to choose between cylindrical, conic or azimuthal projection. Once the goal node is found the most direct route through the graph from the initial state to the solution state is the solution path. The associated state descriptions of each node along this path are then assembled into a solution sequence.

These steps merely describe the major elements of the search process. A complete specification of a search process must also describe the order in which the nodes are to be expanded. If the nodes are expanded in the order in which they themselves were expanded, we have a breadth first search, i.e. each node at a particular level of the hierarchy is tested before proceeding to expand the next level of the hierarchy. Alternatively, if the most recently expanded node is expanded first, we have a depth first search, i.e. we search all levels of the hierarchy on one limb before proceeding to other limbs. Having explored one limb without success the process of 'backtracking' is used to return to a node which has remaining unexplored limbs, thus the system must use pointers to facilitate this process (Figure 1.2).

> Breadth-first and depth-first methods can be called blind search procedures since the order in which nodes are expanded is unaffected by the location of the goal.
>
> (Nilsson, 1971; 43)

The blind search methods are exhaustive measures for finding the goal node (solution), but often they are not feasible because the search will expand too many nodes before a path is found. Since there is always some limit on the amount of time and storage available to expend on search, some more efficient methods of search are required. Also these 'brute force' methods do not allow the use of additional knowledge about the solution to influence the search and clearly are not exhibiting 'intelligence'. If however some information about the global nature of the problem (graph) and the general direction of the goal is available then this may be used to 'pull' the search towards the goal by causing the most promising nodes to be expanded first. It is this use of knowledge that differentiates AI programs from conventional programs. In other words, the key to intelligence is to do as little work as possible. As Forsyth and Naylor (1985; 138) state " ... one method of search is said to be more 'intelligent' than another if the former examines fewer potential solutions than the latter, but still succeeds."

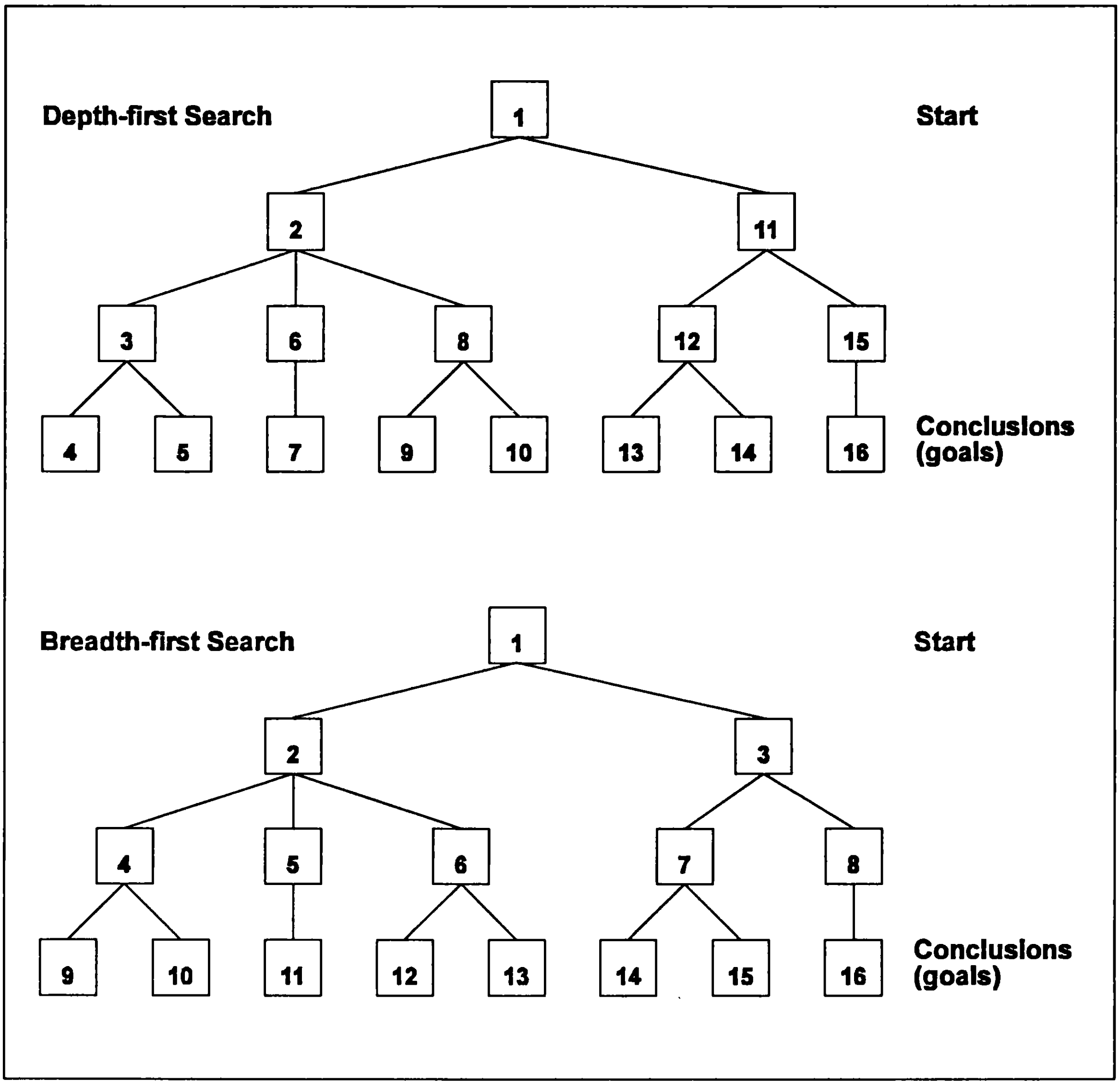


Figure 1.2
Order of opening nodes in depth-first and breadth-first search.

The resultant search strategy is known as a 'best first' search. Here the additional knowledge about the task is used to evaluate each of the open nodes and modify the choice of which successor node to select for further examination. In this way the depth first search expands next the successor thought to be best, or which is most likely to move the search towards the goal state (i.e. provide a solution). Typically, this technique uses 'rules of thumb' or 'heuristics' to evaluate the alternatives. Heuristics are discussed in more detail below in relation to expert systems.

Even if heuristics and their associated evaluation procedures cannot determine the most promising route to a solution, they should at least allow 'pruning ' of the possibilities, that is, those nodes which obviously will not lead to a solution can be 'pruned' so that attention may be focused on those routes which may provide a solution. This does present one drawback: it is possible that a solution might exist but heuristic search may fail to find it, although this is unlikely.

Again to take the map projection example, if the projection is required to show Malaysia the system may suggest a conic projection based upon simple rules of choice, whereas in fact an oblique cylindrical projection may be a better choice.

Another approach to problem solving in AI is that of 'problem reduction'. The basis of this method is to "... reason backward from the problem to be solved, establishing subproblems and sub-subproblems until finally, the original problem is reduced to a set of trivial problems" (Nilsson, 1971; 80). This approach uses 'problem reduction operators' to transfer the 'problem description' into subproblem descriptions. This is similar to the approach taken in conventional programming where a task is broken down into subprograms and subroutines, each performing some small part of the overall task. However, as Nilsson states:

> For any given problem description there may be many reduction operators that are applicable. Each of these produces an alternative set of subproblems. Some subproblems may not be solvable, however, so we may have to try several operators in order to produce a set [of subproblems] whose members are all solvable. Thus the problem of search appears again.
>
> (1971; 80)

Knowledge representation and search procedures are basic to all aspects of AI programming. Only the fundamentals have been discussed here. Obviously there are other representations and search procedures. More detail of these and how they may be applied in practice will be discussed later, particularly in relation to expert systems.

EXPERT SYSTEMS

What are expert systems?

Cynics often seem to view an expert system (ES) as a program consisting of IF THEN ELSE statements and having no other special property, an ES simply being a program with a very large number of such statements compared to conventional programs. Put simply however, an expert system is a computer program which, by using facts and rules about a domain (problem), simulates the decision making process normally carried out by a human expert. They differ from conventional 'algorithmic' programs in both structure and operation.

There are in fact a number of different definitions of expert systems, and what separates them from conventional programs. For example Gero defines expert systems as:

> . . . intelligent computer programs which use symbolic inference procedures to deal with problems that are difficult enough to require significant human expertise for their solution.
>
> (1985; 396)

The British Computer Society's Committee of the Specialist Group on Expert Systems has adopted the following definition of an expert system which emphasises their programming, but allows for a wide range of applications:

> The embodiment within a computer of a knowledge-based component from an expert skill in such a form that the machine can offer intelligent advice or take an intelligent decision about a processing function. A desirable additional characteristic, which many would regard as fundamental, is the capability of the system on demand to justify its own line of reasoning in a manner directly intelligible to the enquirer. The style adopted to attain these characteristics is rule-based programming.
>
> (Simons, 1985; 126)

It may seem that any computer program that solves a problem may be termed an expert system, but there are numerous points which distinguish an expert system from a conventional program, for example

> 1. There is continuous interaction with the user, who conducts a dialogue with the system, and leaves with an answer or conclusion.
>
> 2. The system weighs up the likelihood's, explores alternatives and follows a course of reasoning which depends on the user's replies. Whole areas of investigation may be initiated or discarded as a consequence.
>
> 3. Uncertain or incomplete evidence is accepted and used.
>
> 4. The system elaborates [on] and explains why questions are asked, and describes how conclusions are reached.
>
> 5. Only significant questions are asked, and questions related to a particular topic are grouped together.
>
> (ICL, 1984; 1)

Put simply, in a conventional program the user follows a rigorously defined series of steps to meet the requirements of the program exactly. In an expert system, the interaction is flexible and should emphasise the requirements of the user.

The most comprehensive definition of expert systems is probably found in Hayes-Roth et al. (1983), but according to Merry and Hammond in their report on the first Alvey Directorate workshop on expert systems, the term 'expert system' has in fact proved most difficult to define, and " ... most short pithy definitions usually exclude computer programs which one would consider to be expert systems, and include those that one would not" (Merry and Hammond, 1984; 1). In any event, a true expert system should rival the performance of human experts.

The term expert system has become much used and abused in recent years and is probably best now used as a general term covering a small number of specialist program types able to use facts and rules about a subject, infer things from them, and solve a problem or draw some conclusion. It also implies a particular type of structure within the program. In order of sophistication the sub-types of expert systems are perhaps best referred to as Rule Based systems, Knowledge Based systems (KBS) and Intelligent Knowledge Based systems (IKBS).

Rule Based systems, sometimes referred to as production systems, are the type of system most frequently described in the popular computer press of the 1980s as Expert Systems. Most are of the classification or diagnostic type (see below). These systems typically have a series of IF THEN ELSE type rules coded into the program and ask the user to provide some facts in answer to questions from the system. Typical of these are systems which will identify plants or animals in response to information about their characteristics, or fault diagnosis for, say, why a car won't start. These tend to be unsophisticated systems and cannot really be considered to be 'intelligent', although some do have the capacity to increase their knowledge by example. Despite their limitations they are an important phase in the developing field of expert systems.

Knowledge Based systems may be considered to be the next stage of development. In addition to storing rules and facts they may be able to use a variety of structures for coding and interpreting knowledge. Almost invariably the 'knowledge' will be separated from the actual program, they will have the capability of learning from experience, and will apply heuristics or 'fuzzy logic' to problem solving.

Most true experts systems currently fall into this category.

Intelligent Knowledge Based systems (IKBS) is a term sometimes reserved to describe the nebulous future systems we are working towards (Merry & Hammond, 1984; 1), however they have been defined as:

> ... semi-intelligent systems for carrying out a single complex task. This implies working with large, incomplete, uncertain and rapidly changing knowledge store, use of inferential procedures for applying this knowledge in reacting to variegated and unreliable inputs, and the use of sophisticated and flexible control mechanisms.
>
> (Sloman, 1984; 19)

Sloman lists (and discusses) twelve requirements for a system to be considered intelligent. These are: Rich stores of domain specific knowledge; Powerful and varied descriptive resources; General and specific inference procedures; Self monitoring; Meta-principles; Strategies for controlling search; Matching and describing; Communication between sub-systems; Very large very fast memory stores; Rapid re-organisation of part of memory; Fine-grain and coarse-grain parallelism; and Interrupt mechanisms. Some of these requirements are currently incorporated in expert systems, but without access to very special hardware and software it is unlikely that all these requirements can be met. This is reflected by the fact that most current ES are cause/effect driven versus the true inferential systems of the future.

Due to the various definitions of the types of systems discussed above and conflicts as to the hierarchy of systems, in this study the term 'expert system' (ES) will be used as the broad term covering the field implying a system that uses some form of facts and rules to make decisions. Terms such as 'rule-based system', 'IKBS', etc. will be used in the narrower sense described above. For example, 'knowledge based system' will be used to refer to systems where the knowledge base is separated from the inference mechanism, but where the inference mechanism does not have the broad range of 'intelligence' required of a true IKBS.

Inference/learning systems stand somewhat separately from the hierarchy of expert systems discussed above. Most existing expert systems are based upon knowledge obtained from a human expert. The 'knowledge engineer' works with the expert to obtain domain specific knowledge and organises it for use by the program. The expert is called upon to perform a difficult task, with which he is also unfamiliar.

> He must set out the sources and methodologies of his own expertise, and do so in such a way that it makes sense to a non-expert [the knowledge engineer] and can even be represented in a precise machine readable form!
>
> (Quinlan, 1982; 193)

This frequently is a difficult task and often creates a bottleneck in the development of expert systems.

It is possible that machine inductance can replace the traditional knowledge engineer to some extent. As explained by Berry & Broadbent:

> In this technique large sets of examples from the task domain are fed into the system as raw data and the system applies an inductive algorithm to discover the simplest set of rules which will generate those examples.
>
> (1986; 229)

In other words, 'inductive inference', or learning by example, is a process of going from the particular to the general.

A problem with this approach is that it requires a large database of documented examples, which is not available or possible for many areas of human expertise. Also, " ... the rules induced from examples are often extremely complex and difficult to understand" (Berry & Broadbent, 1986; 229). It may be more appropriate for the expert to guide an inductive inference system in its search for regularities rather than trying to specify the knowledge directly (Quinlan, 1982; 193).

Although one can conceive of independent learning machines, the concept is also seen as being a basic component of all expert systems, in that a system should be able to learn from its experience, and hence be able to solve problems better or faster on future occasions. For this learning to take place there must be some feedback into the system to let it know how it should modify its behaviour (Forsyth & Naylor, 1985: Levine et al., 1986)

The Role of Expert Systems.

An obvious question to ask is why there is a need for expert systems, rather than rely on human expertise. According to Basden, the benefits lie in:

> greater reliability (will not forget factors).
> increases consistency (same importance given to factors).
> increases accessibility.
> the ability to arrive at a faster solution or try a greater number of alternatives in the time available.
> the easier duplication of expertise (less training).
>
> (1984; 61)

In the case of design, increased consistency also implies repeatability, something not always achieved in manual processes. It should also be easier to document and afford artificial expertise, and it is more permanent (Waterman 1986; 12).

Expert systems are especially appropriate where there is no efficient algorithmic solution. "Such cases are called ill-structured problems ... " (Glarranto & Riley, 1989; 20).

There are of course disadvantages to expert systems, hence there is good reason not to eliminate human experts, but to supplement them. Human experts are creative and adaptive and although expert systems can gain through experience, they are not as flexible as humans. Expert systems rely upon symbolic representations of objects and relationships and cannot make use of the wide range of complex sensory inputs available to humans.

> ... human experts and nonexperts alike have what we might call 'commonsense knowledge'. ... Because of the enormous quantity [and range] of commonsense knowledge, there is no easy way to build it into an intelligent program, particularly a specialist like an expert system.
>
> (Waterman, 1986; 15)

A further limitation of many expert systems is their lack of causal knowledge. "That is, the expert systems do not really have an understanding of the underlying causes and effects in a system" (Giarranto & Riley, 1989; 8). They tend to rely upon shallow knowledge such as heuristics rather than deep knowledge, and as Giarranto and Riley point out:

> Human experts also know the extent of their knowledge and qualify their advice as the problem reaches their limits of ignorance. A human expert also knows when to break the rules.
>
> (1989; 7)

Computer systems cannot attach meanings to the data they use. Facts imply that something exists or is true, and these facts can be related by rules, but these are symbolic representations of (part of) the real world, rather than the fuller knowledge and understanding of the world an expert possesses.

Expert system structure

Although some of the superficial differences between expert systems and conventional programs have been discussed, it is perhaps in the underlying, internal structure that the differences are most apparent.

The simplest model of an expert system consists of three main parts (Figure 1.3). These are the knowledge base, the inference mechanism or inference engine, and the user interface. The term user interface is self explanatory, referring to the part of the system that communicates with the user. This will be considered in a later chapter. It is the structural difference whereby the knowledge relating to the problem to be solved is separated from the inference mechanism that differentiates expert systems from conventional programs. Clearly an expert system for producing maps must also access a database of relevant data. The necessary extension to the basic model are discussed in Chapter 2 and a model specifically for a cartographic design system is given in Chapter 7.

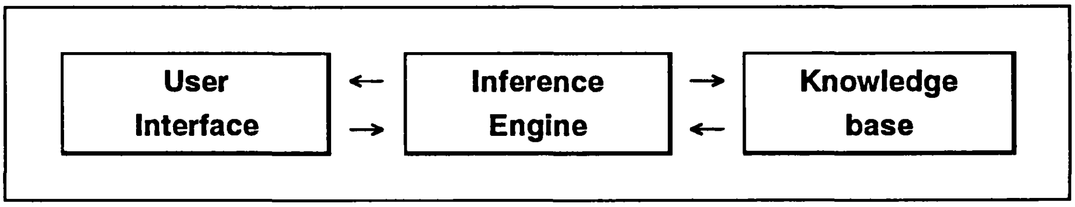


Figure 1.3
Basic components of an expert system.

Expert systems work by relating the contents of the knowledge base to the information supplied by the user's answers to questions formulated by the system. The system infers the most appropriate action in any particular situation, either giving its solution, or asking further questions.

The Knowledge Base. The skill, experience and judgement of one or more human experts is captured in the form of a knowledge base. This can be viewed as a model of the experts' reasoning leading to one or more conclusions. The term knowledge is used by AI scientists to refer to the information a program needs before it can behave intelligently (Waterman, 1986; 16). This information generally takes the form of facts and rules about a particular topic or domain.

Facts are the simplest type of information in the knowledge base. Generally they take the form of some object having a property, e.g. " contours are brown."

Rules are methods or techniques for using (or linking) the facts (Bratko, 1982; 177). They typically take the form IF <condition> THEN <conclusion>. The 'condition' stands for a list of elementary conditions characterising a situation or object to which a rule is applied. The 'conclusion' represents the specific advice or action which this rule indicates when the condition is satisfied (Michalski, et al., 1985; 257). For example, IF two people are brothers THEN they have the same father. Thus, if we have a fact stating that two boys are brothers we can use this rule to infer that they have the same father. The conclusion part of a rule in the knowledge base may be the assignment of the status 'true' to some condition which is in the 'condition' part of another rule. Consequently, the satisfaction of one rule may lead to the satisfaction of another rule, etc., and in this fashion the system can perform a chain of inferences.

The Inference Engine. This is the component of the system which controls the order in which the knowledge base is used, generates new facts from existing rules and known facts (Guilfoyle, 1987; 9), and is generally seen as the central module in an expert system. In Rule Based systems it is sometimes called a rule interpreter. It is in effect the component that provides the system with its thinking power (Simons, 1984; 138). To explain further:

> The inference engine generates answers to queries to the system by either simply retrieving facts from the knowledge base, or, in the case that the answer to the question is not explicitly stored as a fact in the knowledge base, inferring new facts, which constitute the answer to the query, from the facts [and rules] explicitly stored in the knowledge base.
>
> (Bratko, 1982; 177)

Various mechanisms can be used by the inference engine to solve a problem, but it is the use of inference that distinguishes it from the algorithmic approach of conventional programs.

The inference engine can be of a general nature, capable of working with knowledge bases from a variety of domains (commonly known as a shell) or can be optimised to perform in a particular domain. Most early expert systems were of the latter type, but in some cases, such as MYCIN, a system for diagnosing and treating bacterial infections, the inference engine was later adapted to solve more general problems, this being referred to as EMYCIN (Empty MYCIN).

Explanation facilities. If an expert system is to simulate a human expert it must, like a human expert, have some capability of explaining its reasoning. This feature, further differentiating expert systems from conventional programs, takes the form of explaining 'how' a decision was reached or detailing 'why' a particular question is being asked. This means that the user can ask the system for justification of conclusions or questions at any point in a consultation (Merritt, 1989; 55). Although neither of these two facilities are essential to solving the problem, they can help to increase user confidence in the system, and also help to show up mistakes the system may make.

Essentially, HOW? explains the conclusions which the system has reached and is basically a list of the steps gone through to reach the current conclusion, i.e. it would show the nodes on the shortest path between the initial state and the current state.

> The WHY? facility proceeds in much the same way inasmuch as it can be used to give the current state of reasoning of the system - but the main point about WHY? is that it should be able to say which hypotheses are influenced by the current question.
>
> (Forsyth & Naylor, 1985; 26)

That is, the system should be able to state the basic reason for asking the question, and conclusions that may be drawn from its answer. WHY NOT? may also be included to explain why a given conclusion has not been reached.

Although the explanation is often claimed to be an essential aspect of expert systems, its importance to the user may be overestimated. Typically the user is most concerned with solving the problem, often as quickly as possible.

> Furthermore when the user does want an explanation, the explanation [given by the ES] is not always useful. This is due to the nature of the "intelligence" in an expert system.
>
> (Merrit, 1989; 55)

The difficulty experts have in explaining their knowledge is often quoted in relation to knowledge elicitation, but it appears to be assumed that once the knowledge is in the system, explanation becomes trivial. This may be the case for simple factually based systems, but for problems where more intuition is involved, such as in design, then explanation is problematic even for very experienced experts. If an expert system is to provide useful explanations it would need to do more than simply list the rule applied. Deeper knowledge of the problem may be attached to the rules as annotations to be used for explanation, or an alternative approach would be to code deeper knowledge, sometimes called meta-knowledge, into the system and use this to drive both inference and explanation.

The need for an explanation capability and the depth to which reasoning is explained will also vary with the intended users of the system. Clearly a different kind of explanation would be required in a training system to be used by students than in a decision support system for an expert in the domain in question. The explanation system is invaluable to the system developer(s), in which case it serves a similar purpose to program tracing in conventional programs. If the system does not give expected answers, the expert can use the explanations to assess which rules may be in error.

It is an important aspect of expert systems that they should not simply follow a strict sequence of questions, but that the question that will yield the most useful information towards finding a solution should be asked first. Typically this involves the use of heuristics, probability and evaluation functors.

Heuristics

A traditional program is a list of instructions for giving a sure solution to a problem, or reporting that no solution exists. This is known as an algorithm (James, 1984; 10). If one examines the way in which humans solve problems then one sees that very often an algorithm is not followed, but a lose collection of 'rules of thumb' that seem to work are followed. While these rules often do not guarantee a solution, they make it more likely that you will get closer to one.

> A rule that tends to get closer to a solution is known as a heuristic and while it might seem . . . that a heuristic is a 'second class' algorithm , this is far from the truth! Heuristics may not be able to guarantee you a solution to a problem, and they cannot tell you when a solution doesn't exist, but they can be used in a wide range of situations.
>
> (James, 1984; 11)

More explicitly, heuristics are:

> ... criteria, methods or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal. They represent compromises between two requirements: the need to make such criteria simple and, at the same time, the desire to see them discriminate correctly between good and bad choices.
>
> (Pearl, 1984; 3)

Furthermore, when heuristics do produce a solution, it can take far less time than an algorithm would take for the same problem, as was illustrated by best first rather than brute force search methods.

Finding an heuristic may still be a difficult task. The sort of heuristics that humans use are often difficult to discover and difficult to express. James (1984; 11) makes the point that because computers work so fast it is " ... easier to find simple heuristics and allow computers to apply them repeatedly or in very clever ways", rather than searching for a more complex rule.

Thus, although with heuristics the search effort can be greatly reduced, this is at the expense of giving up the guarantee of finding the minimum cost path to the solution for some problems. Practically, the requirement is to minimise the cost of the path and the cost of the search to obtain it (Nilsson, 1971; 54).

In an expert system the use of heuristics for the pruning of the number of possibilities of search to find a goal is known as an 'heuristic search mechanism' (Levine et al., 1986; 22). The heuristic search mechanism focuses attention on the path most likely to provide a solution.

Evaluation functions. Any search procedure other than blind search relies upon some measure by which to evaluate the best path to a solution and becomes an ordered search. Such measures are called evaluation functions.

> The purpose of an evaluation function is to provide a means of ranking the nodes that are candidates for expansion to determine which one is most likely to be the best path to the goal.
>
> (Nilsson, 1971; 54)

There is a variety of ways of applying evaluation functions. Some are based upon the probability that a node is on the best path. Often in board games or puzzles a configuration is scored on the basis of those features that it possesses that are thought to relate to its promise as a step towards the goal. Most complex problems require the evaluation of an immense number of possibilities. "Heuristics play an effective role in such problems by indicating a way to reduce the number of evaluations and to obtain solutions within reasonable time constraints" (Pearl, 1984; 4).

Certainty factors and fuzzy logic. Facts and rules in expert systems are not always either true or false. Sometimes there is a degree of uncertainty about the validity of a fact or the certainty of a rule. When this doubt is made an explicit part of the knowledge base it is called a 'certainty factor' (Waterman, 1986; 16). This certainty factor is usually expressed as a number between 0 and 1, 1 representing total certainty and 0 representing maximum uncertainty. A value of 0.5 would mean that a fact or decision may be correct only half of the time. Facts and rules in such a system could be expressed as follows:

> FACTS: Building 3047 contains tank No23 with certainty 1.0
> : The power saw was defective with certainty 0.8
> RULES: If the spill material is sulphuric acid with certainty 1.0 then
> the source of spill is building 3047 with certainty 0.9
> : If the product was defective with certainty > 0.5 the theory
> of strict liability applies with certainty 1.0
>
> (Waterman, 1986; 16)

Thus, the rule format permits you to express the conditional knowledge of experts and also the confidence or lack of confidence the expert has in this knowledge (Michalski, et al., 1985; 257).

What this means in terms of expert systems is for example that based on a patient's symptoms a medical diagnostic system will predict that the patient has a particular disease with a given certainty factor; or that based on the geological structure of an area a system may suggest drilling for oil in a certain location and predict the likelihood of finding oil, but it may be a dry hole. In a cartographic expert system a fact such as 'water is blue with certainty 0.8' would imply that water is normally coloured blue, but there may be situations where it is another colour, e.g. on a temperature map where blue is used for cold zones.

There are different methods of evaluating these certainty factors, but they are all based upon an area of probability known as 'conditional probability'. These theories were developed by Bayes and are sometimes referred to as 'Bayesian Probability' (Levine et al, 1986). It is not intended here to get involved in a detailed discussion of probability theories, but rather to give examples of how they may be applied to expert systems.

For many facts and rules, and certainly for heuristics, it is difficult to express any exact mathematical measure of certainty, rather we express ourselves in general terms, such as "it is hot in here." The use of terms such as tall, hot , mild, etc. are all relative linguistic variables that cannot be given a single value. The use of such terms in formulating probabilities is known as 'fuzzy logic' (Levine et al, 1985; 90).

If, in a rule, we have a condition that depends upon two facts each with different certainty factors, it is possible to calculate the resulting certainty that the conclusion is true. This may also be modified by a certainty factory applying to the rule as a whole. The calculations involved can become rather complex, therefore, especially when the certainty factors are based upon rather imprecise linguistic variables, many systems simplify the handling of these calculations. Some systems simply average the probabilities or express figures for the maximum and/or minimum certainty of the conclusion being correct.

Applications of Expert System

Expert systems have been applied to a wide range of problems of different types. Their application to some problems is simpler than to others. The widest application of ES has been to classification and diagnostic problems, which generally have relatively simple flows of logic.

Like the subdivisions of AI, there appears to be no agreement on the sub-types of ES. In the simpler classifications, three categories of ES are common, based on the type of problem they address: classification (such as diagnosis of disease); design; and decision support (Bharath, 1985; 65). Generally, however, a more comprehensive classification is used such as that shown in Table 1.1. Of these ten classes, the first three, interpretation, prediction and diagnosis are all classification problems. Planning, monitoring, debugging and repair may also be grouped together under the decision support heading. The final two entries in Table 1.1, instruction and control, can perhaps be considered meta-systems in that they contain aspects of several different categories. Each of the ten categories are outlined briefly below, together with examples where appropriate.

Table 1.1
Generic categories of expert system applications.

Category	Problem Addressed
Interpretation	Inferring situation descriptions from sensor data
Prediction	Inferring likely consequences of given situations
Diagnosis	Inferring system malfunction from observables
Design	Configuring objects under constraints
Planning	Designing actions
Monitoring	Comparing observations to expected outcomes
Debugging	Prescribing remedies for malfunctions
Repair	Executing plans to administer prescribed remedies
Instruction	Diagnosing, debugging and repairing student behaviour
Control	Interpreting, predicting, repairing and monitoring system behaviour.

After Hayes-Roth et al, 1983, p.14
and Waterman, 1986, p.33

Interpretation systems typically infer conditions from observations, for example the use of seismic observations to interpret the geological structure of an area.

An example of such a system is PROSPECTOR (Gaschnig, 1982) which contains rules linking observed evidence of geological findings with hypotheses implied by the evidence. The system uses probabilities for the facts and rules to predict the existence of mineral deposits.

Prediction systems infer the likely consequences of given situations. This includes weather forecasting, estimating global demand for commodities, population predictions, etc. "Prediction systems sometimes use simulation models, programs that mirror real-world activity, to generate situations or scenarios that could occur from particular input data" (Waterman 1986; 34).

Diagnosis systems "... use situation descriptions, behaviour characteristics, or knowledge about component design to infer probable causes of system malfunction" (Waterman, 1986; 34). This category includes medical, electronic, mechanical and software diagnosis, and is probably the widest used type of expert system currently, with some of the best known expert systems, such as MYCIN, falling into this category.

A good example of this type of system are the rule based systems for car engine fault finding. These generally are relatively simple automated versions of the fault diagnosis charts to be found in 'do-it-yourself' car maintenance manuals.

A more comprehensive example is Plant/ds, an expert system for the diagnosis of disease in soybean plants (Michalski, et al, 1985). This queries the user about factors such as precipitation, temperature, condition of the leaves, stem and seeds, leaf spots, etc. It includes two types of diagnostic rules: expert derived, from the formal knowledge of a plant pathologist; and inductively derived rules, obtained by feeding observations from several hundred cases of disease into a general inductive learning system. Each rule or fact has an associated confidence factor, and the system uses three different evaluation schemes depending upon the situation.

Design systems, sometimes called configuration systems, develop configurations of objects that satisfy the constraints of the design problem. That is, they assemble the proper components of a system in the proper way (Giarranto & Riley, 1989; 18). Examples are electronic circuit layout, building design, chemical or similar plant layout, and creating complex organic molecules, the two most popular areas being molecular biology and microelectronics (Waterman, 1986; 35). Many design systems also try to minimise costs or other undesirable features of potential designs (Hayes Roth et al., 1983; 14).

The most frequently quoted design system is XCON (also known as R1), a system developed by Digital Equipment Corporation for the configuration of VAX computer installations (e.g. Waterman, 1986, Williams, 1986). It takes over the job previously performed by technical editors, who examine a customer's order and determine what computer components are required. "XCON has the distinction of being one of the most mature and widely used expert systems currently operating on a commercial basis" (Waterman, 1986; 217). XCON is a rule based system with over 3000 rules which configures systems at a very detailed level.

> For each order it determines necessary modifications, produces diagrams showing the spatial and logical relationships between hundreds of components that comprise a complete system, defines cable lengths between system components, and handles other jobs usually relegated to skilled technicians.
>
> (Waterman, 1986; 217)

XCON performs at a similar level to an experienced technical editor, but typically performs the task in one minute compared to 20 for an editor. It is now a 'mature' system but will never have all the knowledge required to cover all eventualities, thus it will make mistakes, but DEC have found it to be useful, even during its early development stages.

Planning systems plan a series of actions to perform a function, for example, project planning, communications, experiments and military planning.

An example of this type of system described by Waterman (1986; 264-265) is CARGuide, a system to plan routes and help drivers navigate city streets developed at Carnegie-Mellon University. The system calculates an optimum route from known start point and destination, using information about the road network. Once found, the route is displayed in map form on a graphic display. The car's position is updated during the journey and gives directions at intersections. Numerous similar systems have been developed recently, although not all use expert system techniques.

Monitoring systems compare actual behaviour to expected behaviour, for example, monitoring some instrumental readings to detect accidents or problems in production. A major application area for this type of system is the nuclear industry.

Debugging systems prescribe remedies for malfunctions. "These systems rely on planning, design and prediction capabilities to create specifications or recommendations for correcting a diagnosed problem" (Hayes Roth et al., 1983; 15).

Repair systems follow on from debugging systems by developing and administering a plan to remedy a problem.

Instruction systems, in addition to providing education or training in the topic, can analyse the system user's responses and attempt to correct gaps or faults.

> "Typically these systems begin by constructing a hypothetical model of the student's knowledge . . . Then they diagnose weaknesses in the student's knowledge and identify an appropriate remedy. Finally they plan a tutorial . . . to convey the remedial knowledge to the student.
>
> (Hayes Roth et al, 1983; 15)

The inclusion of explanation facilities in expert systems means that any system should be able to help increase the user's understanding of the problem, but instructional systems have the specific goal of achieving this increased or improved knowledge. Their use is not limited to students in the narrow sense, but they have applicability in anywhere instruction is required.

Control systems, the final class, control the overall behaviour of an operation. To do this they must include a monitoring system to asses the current situation, a diagnostic system to determine what has caused any faults, and probably debugging and repair systems to correct faults. They may also include aspects of the other classes discussed above. Frequently, an important aspect of a control system is the ability to predict future events, and take preventative measures. Applications of expert control systems include air traffic control, business management and mission control.

Expert systems for whom?

Expert System can be used by a wide range of people for a variety of purposes. The major groups of users are likely to be experts themselves, practitioners[2], students and those with no experience in the field.

Experts will use expert systems in a decision support role, using them along with other decision support systems to confirm their decisions, or to act as intelligent checklists. The expert system may be used like an intelligent assistant and as more intelligence is added to it, it acts more and more like an expert. "Developing an intelligent assistant may be a useful milestone in producing a complete expert system" (Giarranto & Riley, 1989; 3)

Also, although currently it does not appear likely that ES will replace the human specialist, they will reduce the number of trivial enquiries, thus allow the specialist to devote more time to less trivial problems.

Some expert systems will no doubt be used by novices, but there is some feeling that this will be less widespread than was at first thought. Most fields of study have their own special words or 'jargon' or apply special meaning to ordinary words that a novice might be dangerously unaware of (Basden, 1984; 64). Practitioners on the other hand will be familiar with the jargon of their domain. According to Basden (1984) it is also likely that expert systems for practitioners will be more cost effective than for novices.

Expert systems have much to offer in education. Instructional ES were discussed above, but because of the nature of expert systems generally and their

[2] A practitioner is one who has some experience in a domain, but does not have the deep specialist understanding of an expert.

capability to explain their reasoning, most expert systems will be useful in training. Students, like practitioners, will have some awareness of the 'jargon' and will be able to use expert systems in example cases, or to test their own hypotheses.

A final group who will likely make use of expert systems are specialists or experts in one particular domain wishing to apply their knowledge in a related field, or to make use of a system to process their information; for example a geologist using a cartographic expert system to map his data. He has specialist knowledge about the geology of the area, but not the cartographic knowledge to produce the map. This kind of expert system use receives little coverage in the literature.

The need for such systems for producing maps perhaps has something to do with how many view cartography. Most Intelligent Computer Aided Design (ICAD) systems are directed at assisting designers, not at making it easier for non designers to produce designs.

PATTERN RECOGNITION

Pattern recognition was initially primarily concerned with the study of artificial or computer vision, 'pattern' pertaining to visual pattern. The term has now been extended to include such topics as patterns of sounds, patterns of events, etc. James (1984; 82) notes that: "Most areas of AI use pattern recognition to some extent, but usually in combination with other methods and theories that tend to be just as important."

Perhaps the most obvious application of pattern recognition in mapping is in the recognition of features on remotely sensed images, as part of an image processing system. Generally the problem of recognition can be broken down into two stages, feature extraction and classification as illustrated in Figure 1.4.

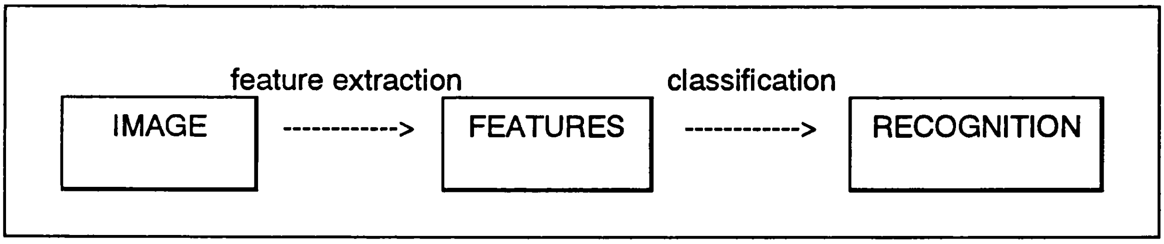


Figure 1.4
Stages of image processing.

Sometimes the recognition of patterns is an end in its self, such as in computer vision, letter recognition or speech recognition, which need not be used in conjunction with any other 'intelligence'.

> These and other important recognition problems have tended to emphasis pattern recognition as a subject in its own right with few connections with the rest of AI. However, it seems reasonable to suppose that this will change as acceptable solutions are found to the simpler pattern recognition problems.
>
> (James, 1984; 99)

Artificial vision and hearing are clearly important to AI, but the other aspects of pattern recognition, although less obvious, are equally important. For example, one of the problems of implementing expert systems is to recognise the 'condition' that forms part of a rule. Also, the recognition of patterns within data or information will aid expert system processing.

For the purposes of this study pattern recognition will not be further considered. However, it is likely to have increasing importance in future mapping systems, not just as part of an image processing system, but also aiding in the solution of a broader range of problems as part of the next generation of AI programs in mapping. Its use can be foreseen in such applications as recognition of information type from user descriptions by providing a best match to existing patterns; the recognition of line 'types' as an aid to generalisation, e.g. recognising different types of coastline; and in the longer term the recognition of distributions and how they may be generalised or classified, and perhaps make a contribution to the difficult problem of assessing map complexity.

NATURAL LANGUAGE PROCESSING

Natural Language Processing (NLP) is that branch of Artificial Intelligence that tries to make the computer able to understand commands written (or spoken) in a standard human language such as English. It is also concerned to some extent with creating computer responses which appear to be in a natural language, but this is less difficult, and follows fairly simply if the first problem is solved.

Natural languages are those which are spoken and understood by large numbers of people: they haven't been invented, but have gradually developed over long periods of time, hence the term 'natural'. They are complex, continually developing and changing, and while there are often rules e.g. grammar, exceptions often exist and it may be difficult to understand the logic in some situations.

Computers are more capable of handling formal languages. These are languages that have been invented and defined, usually for some specific purpose. Programming languages such as BASIC, Pascal, etc., are all formal languages. Because these have a rigid structure and meaning computers are very good at understanding them. Gradually, computer commands and languages have become easier to use and now often resemble English, but usually there are strict limitations on the structure and usage of commands.

Natural language processing has little or no use on its own, but is very important in providing the 'front end' or part of the 'user interface' to other computer programs (Schildt, 1987, 93). The application of NLP will be discussed along with other types of user interfaces in later chapters.

CONCLUSION

This chapter has discussed the divisions and mechanisms of Artificial Intelligence and Expert Systems in some detail. If in the long run computers are ever to help people in general they must: " ... cease to be the preserve of scientists, technologists and programmers and become a universal asset that everyone can get something out of" (James 1984; 121). One of the objectives of current work in AI and ES is to lower the threshold of knowledge necessary to begin using a computer. "To this end it is important that part of the development of AI concentrates on producing flexible systems that can interact with humans to supply and record knowledge (James, 1984; 121).

The following chapters examine expert systems in more detail, look at the application of AI to mapping and detail the development of an expert system for one particular application.

CHAPTER TWO

Building Expert Systems

> The computer as an intelligence amplifier is an abstract idea that we are still a long way from implementing. Today most of the mutual working together of man and machine is on the machine's terms![1]

CONCEPTUAL MODELS

Weiss and Kulikowski identify three stages in the development of an expert system. The first stage is the initial knowledge base design which deals with problem definition, conceptualisation, and forming the computer representation of the problem. The second phase is the prototype development and testing stage and the third involves the refinement and generalisation (i.e. making it more generally applicable) of the knowledge base (1983; 13).

In the development and usage of expert systems we can distinguish three groups of people (Poiker et al, 1982). First, the systems designer or computer scientist, now frequently called the knowledge engineer. Second, the specialist or expert who creates the knowledge base, or whose knowledge is used to create it; and third, the user of the system who may have some expert knowledge of the subject, may be an expert in a related field or may be a student.

The task of the knowledge engineer is to: 1) define the expert system domain; 2) elicit the desired information from the human expert(s); 3) structure that knowledge in a suitable form in the knowledge base; and 4) test the system to evaluate its robustness and accuracy (Williams, 1986; 67).

As noted in chapter one, an expert system consists of three essential components, a knowledge base, an inference engine and a user interface. To this other components may be added, such as a knowledge acquisition subsystem and an explanation subsystem. The latter of these explains why a question is being asked and how a conclusion has been reached; this capability is frequently incorporated into the inference engine. For cartographic design purposes a database of geographic information (spatial and attribute) and a graphical output subsystem must be added (Figure 2.1).

[1] James, M. (1984) Artificial Intelligence in BASIC, p.121.

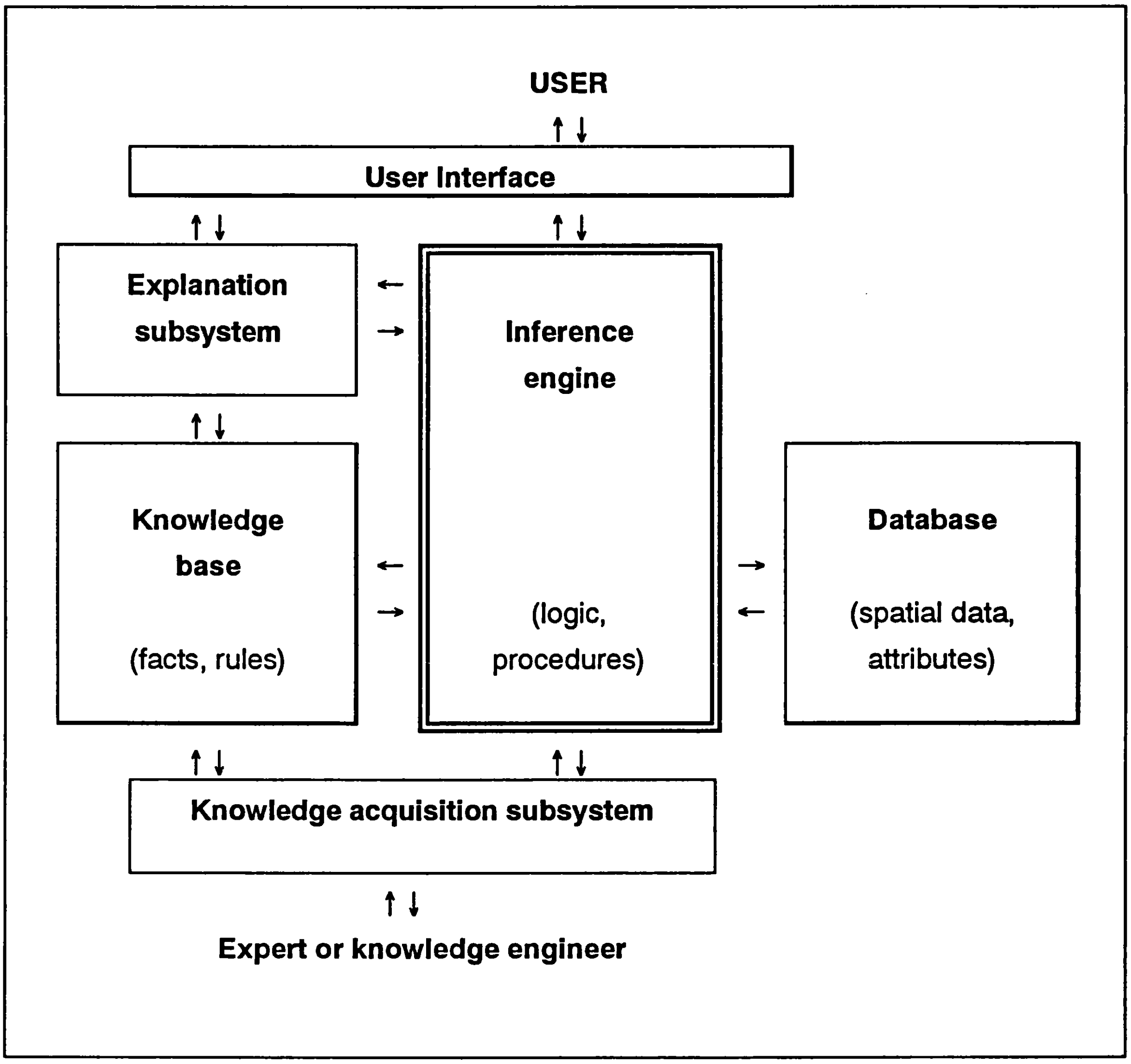
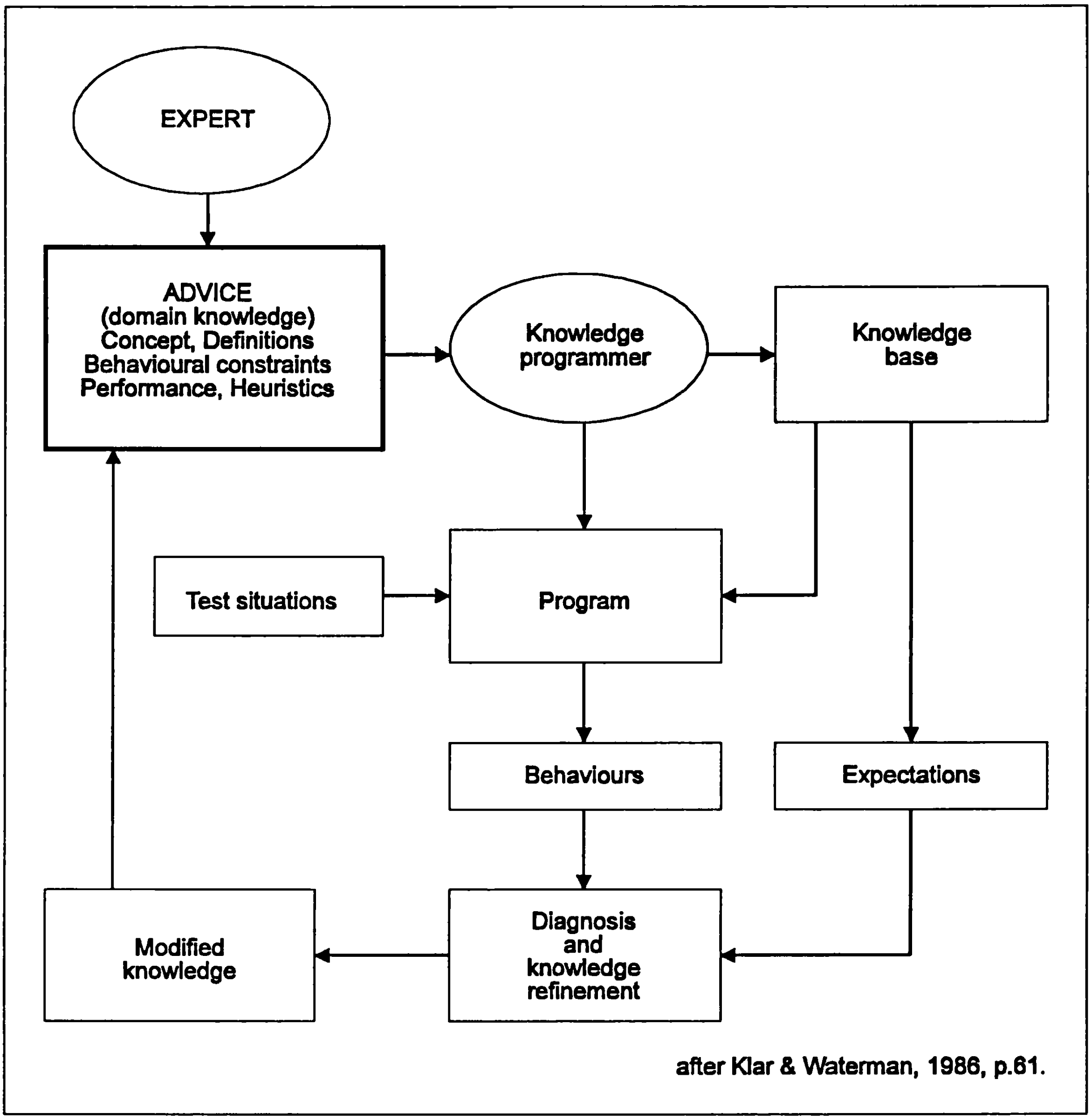


Figure 2.1
Basic components of a cartographic expert system (a more comprehensive model is given in Figure 7.1) (After Harmon & King, 1985)

KNOWLEDGE ENGINEERING

Knowledge engineering encompasses a number of tasks in the development of an expert system. These include problem identification, an assessment of the usefulness of a system, cost-benefit evaluation, locating suitable experts, knowledge elicitation, conceptualisation of the problem, translation of the knowledge into a computer representation, and the testing evaluation and refinement of the system (Weiss & Kulikowski, 1983) as illustrated by Figure 2.2. The process is highly iterative and may result in several prototypes (Lundberg, 1989).



after Klar & Waterman, 1986, p.61.

Figure 2.2
The process of developing an expert system

The development of an expert system is quite different to that of traditional software engineering. For problems where an existing expert system shell or toolkit is suitable, the ability to think rationally and communicate well are more important than ability in computer programming (Williams, 1986).

Problem Identification

The aim of this phase is to determine the goals of the system and recognise the constraints placed upon it. Rabbits & Wright list five aspects to be considered during initial problem assessment. These are: who will be affected by use of the proposed system; what are the success criteria; what are the constraints on the system (time, cost, etc.); what assumptions are made; and what is the scope of the system, especially what will it **not** do (1987; 16).

Knowledge Elicitation.

A number of methods for knowledge elicitation have been adopted (Table 2.1). Despite this range of methods, experience has proved that knowledge elicitation is a major bottleneck in the development of expert systems (e.g. Berry & Broadbent, 1986; 228, Kidd, 1987; vii, Merrit, 1989; 3). Experienced knowledge engineers frequently describe the process as being " ... more of an art or craft than a science" (Berry & Broadbent, 1986; 228). The expert has to be able to explain

Table 2.1
Knowledge elicitation techniques and information that can be obtained

Technique	**Main information types**
Focused interview	Factual knowledge Types of problems Functions of expertise
Structured interview	Structure of concepts Mental models Explanation
Introspection	Global strategies Justification Evaluation of solutions
Observation	Use of knowledge Reasoning strategies
User dialogues	Reasoning strategies Modality information
Review of literature	Factual knowledge Repair of gaps (Re)interpretation of information Support knowledge

based on Breuker & Wielinga, 1987; p23.

the domain for which the system is developed in a manner which is understandable to both the knowledge engineer and the eventual system users. Knowledge elicitation has a reputation for being a difficult and unpredictable process. A reason for this is that none of the commonly used methods are entirely satisfactory (Rabbits & Wright, 1987). It also assumes that the capabilities of the expert are transferable, which may not be the case, particularly if the wide experience and/or intuitive skills of the expert are involved.

One reason for difficulty in knowledge elicitation is, as Sagalowicz points out, that true expert knowledge is not only rare it is seldom explicit or measurable and:

> As a result it is difficult to communicate or acquire. Experts gain their knowledge through experience, long periods of training, apprenticeship and observation. Their value comes not from the number of facts they know but the subtle, idiosyncratic ways in which they come to organise their knowledge and experience.
>
> (1984; 138)

Not only is the knowledge itself difficult to quantify, but experts often cannot articulate how they solve a particular problem, or explain their approach in a systematic manner. Good knowledge acquisition is however, critical, as the resulting expert system is dependent upon the quality of this knowledge (Kidd, 1987; 1).

In the initial stages of knowledge elicitation written documentation can be used, but this is rarely complete and additional information has to be obtained directly from the experts (Breuker & Weillinga, 1987; 21). Interviewing is the most frequent method of knowledge elicitation, although one problem with this method is that the expert may be led in certain directions by the knowledge engineer, which may not result in the best outcome (Rabbits & Wright, 1987; 16).

Introspection is where the expert himself analyses the problem and extracts the necessary expertise. As has been noted, experts often find it difficult to describe their expertise in a systematic manner, but many expert systems have been developed in this way.

In the observation method the expert 'thinks aloud' as he solves a problem. The knowledge engineer records the events, but it is possible for him to misinterpret specialist knowledge, there may be gaps in what the expert says, or the real reasoning may not be apparent even to the expert (Rabbits & Wright, 1987; 16).

Whatever method is used, gradually a large knowledge base of facts and rules will be built up. In testing the system however, it is likely that the expert and the system will disagree at some stage, therefor the expert must either provide a new rule, or modify an old one. " ... sometimes the knowledge the human expert provides is not knowledge about the task itself but, rather knowledge about the knowledge in the program" (Hayes-Roth et al., 1983; 220). This information is known as metaknowledge, i.e., knowledge about knowledge.

In fact Hayes-Roth et al., (1983) recognise three levels of knowledge provided by experts: factual; heuristic; and metaknowledge. The first level provides a base of facts, theorems, equations, categories and operations. This is typically equivalent to the factual knowledge contained in textbooks on the subject. The second level of knowledge is heuristic, including rules of thumb, inconsistent advice and inexact judgmental criteria. This heuristic knowledge provides a "first order correction" to the factual knowledge. This may be further modified by the application of metaknowledge, providing " ... a 'second order correction' of the previous system knowledge" (Hayes-Roth et al., 1983; 222)

Almost all expert systems incorporate all three levels of knowledge, although there may be advantages in explicitly identifying metaknowledge.

THE KNOWLEDGE BASE

With traditional programs 'knowledge' is contained within the program code itself. Moving much of this knowledge to a separate knowledge base has several advantages. It is easier to update the knowledge i.e. to add new rules if, for example, more data is added to the database, or additional map types were required. The actual rule definitions in a separate knowledge base should be more apparent to people other than the original programmer as it is frequently very difficult to follow the exact flow of logic within a complex computer program.

> Abstractly,
>
> . . . knowledge consists of descriptions, relationships and procedures in some domain of interest. The descriptions in a knowledge base, which identify and differentiate objects are sentences in some language whose elementary components consist of primitive features or concepts. A descriptive system generally

> includes rules or procedures for applying and interpreting descriptions in specific applications. A knowledge base also contains particular kinds of descriptions, known as relationships. These express dependencies and associations between items in the knowledge base. Typically such relationships describe taxonomic, definitional and empirical associations. Procedures, on the other hand, specify operations to perform when attempting to reason to solve a problem.
>
> Hayes-Roth et al., 1983; 12

A number of strategies and tactics to enable expert systems to solve problems have been investigated. There are two fundamental aspects to this: how the knowledge is organised or represented; and the method of search for the solution. Obviously these are to a great extent dependent upon one another.

Knowledge Representation

'Knowledge representation' is the term most frequently used for the internal representation of information in the knowledge base. In simple terms it involves writing down, in some language or communication medium that the computer can comprehend, descriptions that correspond to real world information.

As has been discussed, knowledge generally takes the form of facts, rules and heuristics. While this is the most basic and common representation, sometimes known as first order logic, others are possible such as frames and semantic networks. How best to organise the knowledge to support problem solving is an important aspect of AI programming, but at the same time the knowledge in an expert system must be transparent to the human user, easily incremented by a human expert and easily modified by the human expert (Bratko, 1982; 180).

No one method of knowledge representation appears to be particularly efficient in all situations. The problem is that the same piece of information can be used in a number of ways and in a variety of contexts in solving a problem. There are two basic school of thought (Steels & Campbell, 1985). The declarativists believe that information should be represented in a neutral fashion, i.e., independent of its use. Control can be achieved by general purpose problem solving strategies.

Alternatively, proceduralists believe that information cannot be represented without some indication of how it may be used, i.e. the choice of representation necessarily determines the complexity of the processes operating over it. This latter view has gained popularity with the development of object oriented programming.

Here the data structures are such that they contain not only data about objects, but also information on how the objects may be used or operated upon.

First-order logic. This is a formal method of representing logical propositions and the relations between them i.e. facts and rules. This is probably the simplest way to represent knowledge and the most widespread. It has been used extensively in classification type rule based systems.

There is a template for each fact which consists of title, or relation name, followed by one or more fields containing specific values (Giarranto & Riley, 1989; 380). For example:

```
Male (John)
Male (Alan)
```

are two instances of the fact Male which state that John and Alan are male. Relationships can also be expressed by facts:

```
Father_of (John, James)
Father_of (Alan, Louise)
```

Facts can be related by using rules, for example if facts about the sex and parenthood of individuals were stored in the knowledge base a rule could be used to determine if two people are brothers:

```
Brother_of (A, B):-
        Male (B),
        Father_of (X, A),
        Father_of (X, B).
```

This rule states B is a brother of A (not necessarily vice versa) if B is male and both A and B have the same father.

A more easily recognisable way of expressing this knowledge is in the form of production rules, such as:

```
IF:     A is TRUE AND
        B is TRUE AND
        C is FALSE
THEN: Conclude X
```

Semantic networks. These are the most general knowledge representation schema (Harman & King, 1985; 35), and were first developed for use as psychological models of human memory (Waterman, 1986; 70). A semantic network is composed of nodes and links (Figure 2.3). Nodes may represent objects, concepts, situations, or descriptors. The links (or arcs) describe the relationships between nodes, and may be directional. Links may be affirmative such as 'object is-a member of a class' or 'object has-a property', or they may include heuristics, such as 'situation may-cause description'. The relationships between objects in network systems can be more varied than in hierarchical systems. According to Simons (1983; 136) the simplicity with which correct deductions can be made once the semantic network has been generated is one of the main reasons for their popularity. They have been particularly successful in natural language systems for the representation of complex sentences (Waterman, 1986; 72).

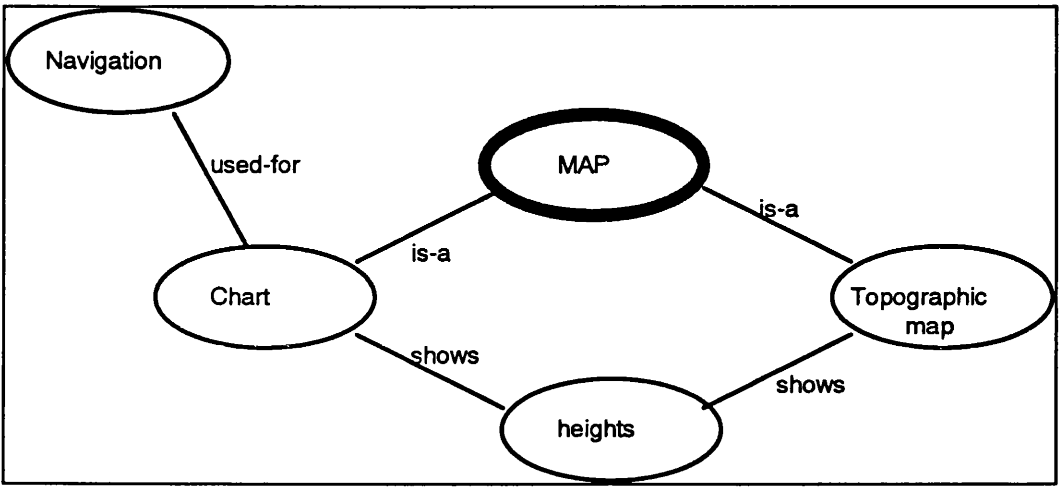


Figure 2.3
A semantic network

Frames. Essentially these are semantic networks in which the knowledge is represented in modular chunks rather than as individual items (Simons, 1985; 136). In its simplest form a frame is like a questionnaire, consisting of a series of items, ' ... each of which has a specific purpose and each of which has an associated blank which must be filled to get the complete picture" (Forsyth & Naylor, 1985; 134). A frame can contain several different types of information:

> Some of this information is about how to use the frame. Some is about what one can expect to happen next. Some is about what to do if expectations are not confirmed.
>
> (Waterman, 1986; 73)

Each of the blanks or 'slots' in the frame must be filled in order (Figure 2.4). This is achieved by a procedure or procedures being associated with each slot which may for example, ask the user to answer a question, or refer to further frames, thus resulting in a hierarchical system in which the topmost frames represent generalities and the lower ones may be customised for more specific instances or concepts by the creation of more specific frames. In other words:

> Frames attempt to model real world objects by using generic knowledge for the majority of an object's attributes and specific knowledge for special cases.
>
> (Giarranto & Riley, 1989; 83)

A basic characteristic of a frame is that it represents related knowledge about a narrow subject, particularly when there is a considerable amount of default knowledge. According to Giarranto & Riley " ... frames provide a convenient structure for representing objects that are typical to a given situation such as stereotypes" (1989; 82). For example, many maps have several similar characteristics and in many cases default values are adequate. Further details of the components of the feature (map) can be obtained by examining the structure of the frame.

Frames are also useful for representing common-sense knowledge which is generally difficult to handle in computer system. While semantic nets are better for representing broad knowledge, the advantage of frames is the ability to build hierarchical systems with inheritance. "By using frames in the filler slots and inheritance, very powerful knowledge representation systems can be built" (Giarranto & Riley, 1989; 83).

Frame:	**MAP**		
slot name	**value**	**if empty procedures**	**on change procedures**
topic:		menu of known types	
map class		derive from map type	
date		default = system date	
map purpose	[overview]	choose from menu	update level of detail
map user	[general]	choose from menu	update level of detail
output media	[screen]	choose from menu	update level of detail
level of detail		derive from purpose, user and media	

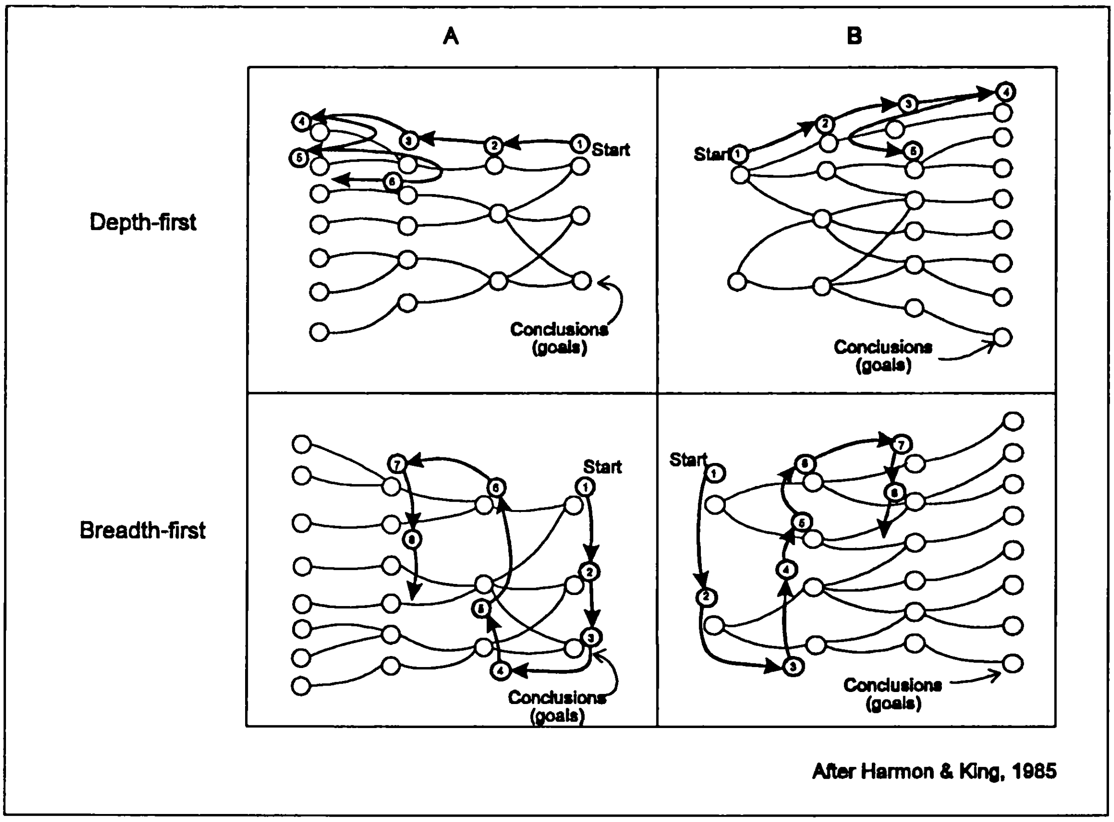
Figure 2.4
An example of a frame for a map. The attached procedures are used to get values for slots ([ ] = default value).

Blackboard. Often an important feature of problem solving is that diverse types of knowledge must be handled. This may mean that more than one expert system is needed. The communication of information between expert systems is done through the blackboard mechanism. This is an area of computer memory (or storage) where information stored within an expert system can be 'posted' in a structured form so that it can be accessed by other expert systems if required to reach their goals (Levine, et al., 1986; 23, Waterman, 1986; 146). Some systems also use a blackboard for storing intermediate results (Hayes Roth, et al, 1983; 16).

THE INFERENCE ENGINE

Mechanisms of expert systems

In any complex process of reasoning a whole series or 'chain' of rules may have to be considered. The relationships typically form a hierarchical tree structure, thus the procedures used are typical of the graph search methods outlined in chapter one, although there are two distinct approaches used in expert systems. These are known as forward chaining and backward chaining. Both of these methods can be applied to breadth first and depth first search, as illustrated in Figure 2.5.



After Harmon & King, 1985

Figure 2.5
Major categories of search strategies used by inference engines

Forward Chaining. Analysis by continually narrowing down the possibilities is called 'forward chaining', and is so called because the information is considered before the conclusions, which is the same order as used in writing the rules (Davies, 1986; 6). This works like inductive logic by working forward from existing facts and rules to derive new ones that are true (Williams, 1986; 69), but must also consider the various search possibilities at a given node.

Forward chaining, sometimes called a data driven strategy, is simple to program, is much used in rule based systems, and appears much as if the computer is working through a list of possibilities, making what inferences it can from the answers. It is quite likely with this method that a wrong line may be taken, resulting in backtracking to find the correct path, and giving the appearance to the user of an unconnected series of questions. More sophisticated systems make better use of heuristics and evaluation functions in an attempt to avoid this.

This strategy is most appropriate where many facts are known and the search tree is broad but not deep.

Backward Chaining. The converse situation occurs when one starts with a conclusion and works back to find out if the conditions for that conclusion are true (Guilfoyle, 1987; 9). This is known as backward chaining, or a goal driven strategy, and like deductive logic one works back from a given hypothesis or goal, searching the knowledge base for facts and rules which support (or disprove) it.

> According to Bramer, backward chaining :
> ... has the additional value that it helps to ensure that groups of questions asked by the system appear 'focused' towards evaluating a particular hypothesis. Once a hypothesis is found to be justified or refuted, further questions relating to it do not need to be asked.
> (1982; 14)

This produces a form of information gathering that is probably more acceptable to most users than the more random gathering of facts by forward chaining, and is widely used by many of the well known systems such as MYCIN. Backward chaining is best applied where there is a known hypothesis for which evidence can be gathered. " ... backward chaining facilitates depth first search. A good tree for depth first search is narrow and deep" (Giarranto & Riley, 1989; 164).

In fact, it is possible to combine forward and backward chaining in one system, using them at different stages in the search for a solution. For example in a frame based system, many slots may be filled by assembling facts from the knowledge base or user and moving forward to more specific frames. However, some slots may contain hypotheses (or default values) for which backward chaining could be used to determine their validity.

Although an expert may be able to visualise the final map and break this down to produce specifications, it is difficult to see how an overall backward chaining strategy could apply in a map design system as neither the non expert nor system can easily hypothesis a map design then test it. This is due partly to the difficulty of evaluating good design as shall be discussed later. It is quite likely however that backward chaining would be used in certain stages, for example by assigning some default representation method to a particular phenomena, then testing for other factors such as scale, map purpose and the symbolisation of other phenomena to confirm or reject this representation.

Expert System Shells

An expert system shell is in effect an expert system without any built in knowledge base. That is, it consists of the inference engine and an empty knowledge base. The theory is that once an inference mechanism has been developed it can be applied to solving more than one problem. Many expert system shells exist, one of the most famous being EMYCIN or Empty MYCIN, the MYCIN medical diagnostic system with the knowledge base removed. Clearly for an expert system shell to be of use, the inference mechanisms and knowledge representation methods used must match those of the problem to be solved.

The VP Expert shell has been applied to some cartographic tasks, (Siekierska 1989), but generally it has been found that the inference mechanism of commercially available shells does not support the requirements of cartographic systems, and until recently few have the necessary graphical capabilities.

THE USER INTERFACE

"From the very earliest times Man has been trying, without much success, to speak to, and receive intelligible replies from, non-human objects" (Forsyth & Naylor, 1985; 35). It seems natural therefore that one should be able to converse easily with a computer. The major reason that this is not the case is essentially the problem of natural language. Natural language is what we speak and write, and although there are many rules, there is no language definition, it has just developed (naturally) with time. The same cannot, however be set for the other type of language, formal language.

Formal languages are those which have been designed for specific purposes, the most common examples being computer languages. Inherent in these is some formal definition of the language. So, despite advances in the man/machine interface, making computers, and more particularly certain computer programs, easier for users to communicate with, the formal language will still remain which will require the user to learn at least some rules of the language.

In order to communicate in any language the recipient of a message must be able to understand it. Models of communication have been extensively used in cartography in recent years, but many apply equally to (or were copied from) other areas of communication research. Thus, any language may be viewed as a communication chain such as:

message > encode > transmission > decode > message

For communication to succeed then, the message at the recipient must be the same, or at least have the same meaning, as that at the sender.

When the computer has a message it will print this on the screen. (For the current discussion we will ignore the possibility of acoustic communication). How clear this message is will depend upon the individual program, but generally speaking it is not very hard to make the computer's responses appear to be in natural language, mainly because in any given situation the number of messages that the computer is likely to want to communicate will be limited. Careful programming should ensure that messages are unambiguous and easy to interpret. An example of this may be the computer giving an error message describing an error condition which exists, rather than simply giving the error number. Although in the later case communication may still be successful, the user will probably have to find the appropriate manual and look up the error number to get the meaning of the message.

There is also the advantage that when it is the computer which is doing the sending, it is up to the user to understand the message. The brain has a much greater processing power for this type of problem than the computer, so in the difficult stage of decoding the message is passed to the more powerful processor, with the easier process, encoding, being handled by the less powerful processor (the computer) (Forsyth and Naylor, 1985;39).

While some form of natural language communication between the computer and the user may be seen as a long term goal, most programmes are very restricted in the language or 'interface' they use. Also, until relatively recently there have been no standard user interfaces, and although the Macintosh desktop, Windows, etc., have promoted standardisation, it often seems to the user that every program has a different interface. Despite this lack of standards, one can identify several generic types of user interface (particularly for input to the computer). These are:

1.	Parameter lists	
2.	Command language input	Strict format
3.	"	Flexible format
4.	Constrained language	
5.	Natural language	
6.	Menu systems	Strict sequence
7.	"	'Random' Access
8.	Graphical interface	

Parameter lists. The commonest form of command and data entry into many early programs relied on a very strict format. Commands, parameters etc., had to be located in a particular column on a punched card or on the screen. Typically everything was numeric, the user referring to a manual to locate the correct sequence and position. Unless one uses such a system frequently it is very difficult to remember the correct positioning of items, even if one can remember all of the various permutations. Errors occur frequently due to what may be considered minor mistakes such as getting a decimal point in the wrong column or a number in the wrong field, or using a decimal number instead of an integer.

These types of program tended also to lack any significant level of error checking, so when an error was present either the program would crash, or the output would contain errors, which may not be easily detected by the user.

With the development of interactive computing the system can prompt for values to be entered, although the strict sequence of questions is maintained. This simplicity can be useful for occasional users, but frequent users may find it tedious to have to read questions that he has come to know.

Command languages. A slightly more user friendly variant of the above, although still designed for batch processing, is the ability to enter a command followed by its parameters. This has been used in several mapping programs, one well developed example being Surface II. This method apparently allows greater flexibility than only entering parameters, but the main advantages are in the readability of batch files at a later date - the command name making it more obvious what the intent is - and the ease with which it can be used interactively, the user selecting commands at will.

Later variants of this approach allow considerable flexibility. One of the better developed versions is the General Purpose Input System of the GIMMS package (Waugh, 1980). Here there is considerable flexibility in the order in which commands are given, the parameters may be entered in a standard sequence, or the order varied by using the appropriate keyword. It also allows comments to be added, which improves understanding if the file is retained for later use.

There are disadvantages in most command systems. Typically the user is faced with a blank screen and must know what command to enter. Often systems develop to a stage where there is a large number of possible commands and options within these commands. Therefore unless the user frequently uses the system repeated reference to manuals or help screen will be essential. Some programs use function keys with templates, or lists of commands in a reserved area of the screen. In the latter case, to preserve space the commands are often abbreviated which may cause difficulties for infrequent users, e.g. the two or three letter cryptic abbreviations used by MapData. In order to avoid ambiguity, particularly when abbreviations are used, the terms adopted are often not the most obvious or meaningful.

Constrained language. In several systems an attempt has been made to make command input appear like English sentences, but there is a strict underlying structure which must be adhered to. One well developed example of this is the MAP system (Tomlin, 1980). This uses an interesting input structure which can operate in either batch or interactive mode. A command is string of alphanumeric characters read as one or more eighty character input lines. The system automatically echoes the input to acknowledge its receipt (the original system was developed for teletype terminals). The command is then either confirmed, an error message issued, or a request for supplementary information made. For example:

```
       prompt    >
      command    protect thismap
         echo    PROTECT THISMAP
error message    NO, THERE IS NO MAP IN THE FILE CALLED
                 "THISMAP"
       prompt    >
      command    rename thatmap to thismap
         echo    RENAME THATMAP TO THISMAP
 confirmation    OK, "THATMAP" HAS BEEN RENAMED TO "THISMAP"
```

(Tomlin, 1980; 21)

Each command must begin with an imperative verb which names the operation to be performed. Unless a command begins with one of these verbs an error will be signalled. In some cases this is all that is required, however, many imperative verbs must be followed immediately by the object of that verb. This imperative phrase may be followed by one or more modifying phrases. Each modifying phrase is made up of a modifier and one or more objects. The order of these modifying phrases is flexible. For example:-

```
AVERAGE THISMAP  TIMES 20  PLUS THATMAP  TIMES 80
```

The command processor creates words from the input line. A blank or blanks must separate each word or number, all non blank characters are assumed to be intentional parts of the input. In most cases the full verb does not have to be supplied, only enough of the first few letters to distinguish it from all other verbs. This has the advantage that longer verbs do not need to be typed in full, but it can lead to confusion on the part of the user (see above). With the commands currently available in the system up to five characters may be required, although often only one or two are needed.

In attempts like this to save keystrokes where abbreviation rules are not uniform, the savings tend to be lost by the user having to remember the right abbreviation for each command (Ledgard et al, 1981; 4). An additional problem arises if the system is later expanded and new commands added. For example if a user becomes used to typing W for WRITE as this is initially the only command starting in "W", confusion could well result if a command WHERE were later added. A more sophisticated system would allow one to type in a possible abbreviation of a command, but if this is still ambiguous, to prompt for more input without the whole input line being ignored as being an error.

Natural language. Several natural language interfaces to database systems have been developed in an attempt to make the systems more user friendly. To communicate effectively between users and systems a natural language interface must have a knowledge about the domain as well as linguistic knowledge (Ishikawa et al., 1987). A natural language interface uses a linguistic model as a knowledge base for semantically interpreting user queries. Domain specific knowledge is required to resolve ambiguities that queries may contain.

In order to ensure that the computer has correctly translated the natural language request it normally must restate the query for approval by the user (Gittins, 1986; 28). This may seem to neutralise the advantage of using natural language in the first place, but is essential due to ambiguities present in most natural languages (and perhaps in the query itself) and confirms with the user the intention of the query before processing the request.

It seems unlikely that general purpose natural language interfaces will be available in the short term. Moreover, there is currently considerable debate within the GIS community about query languages for spatial queries (e.g. Egenhofer & Frank, 1988, Menon & Smith, 1989, Mainguenaud & Portier, 1990, Raper & Bundock, 1990). There does not seem to be any consensus on the how queries should be formed and most of the existing database query languages do not have the ability to handle spatial queries.

Menu Systems. There are numerous ways in which menus can be applied. Menus can be usefully incorporated into command driven systems where the options available for a command are displayed when the command is entered. In this case there is no hierarchy to the menus and one cannot switch from one menu to another without going through the command prompt.

Many early (and current) menu driven systems use a strict sequence of menus in a hierarchical tree structure with the user selecting the appropriate item from a menu to get to the next level menu and so on until the desired result is achieved. To proceed to the next task it may be necessary to traverse back to the top level menu and back down some other branch. Unless the menu is exceptionally well designed it is likely that this task will have to be repeated frequently leading to frustration on the part of the user. Some systems allow shortcuts to be taken, for example by specifying the menu one requires, but this presumes frequent use of the system. At the very least the user should be able to return directly to the top level menu.

More recent menu systems make use of pop-up or pull-down menus. Here the top level menu is available all the time. To access a menu function a 'trigger' key pressed activates the menu. The selection is then made by using the cursor to select the appropriate sub menu and function. To speed up responses for regular users it may be possible to follow the trigger key with a code for the desired function. The widespread availability of the mouse has considerably increased the utility of such systems, although constant changes from mouse to select items and keyboard to enter information can be aggravating.

Menu systems are probably most appropriate where there is a limited range of options available at any given point in operating the system, otherwise the user can waste time searching the available menus for the desired function. They do, however, have the advantage of not requiring users to remember the names of commands.

Graphical Interfaces. These have become increasingly popular in recent years particularly since the advent of the Apple Macintosh computer which relies heavily on them and the increasingly widespread use of Microsoft Windows on PCs. They are based on the use of a mouse to interactively select icons or item from menus. Menus or commands are selected by pointing to them with the mouse pointer on the screen and clicking the appropriate mouse button. Further sub menus may 'pull down' or 'fly out' to allow further choices or 'dialogue boxes' may appear to allow the user to enter further information.

While these interfaces are supposed to be intuitive and are supposed to imitate the user's desktop, they often pre-suppose some familiarity with the system although the fact that the main menu headings and most commonly used icons are always visible means that less reliance has to be placed on memory than with command language systems. The speed of moving from one command to another also gives them distinct advantages over structured menus in many situations.

Perhaps the greatest advantage of these systems is that fact that all programs conforming to the Windows convention, for example, will have a similar 'look and feel', i.e. the user interface to a wide range of programs is essentially standardised.

A disadvantage of these interfaces is that casual users can be left wondering what to do next, or where to find the appropriate command.

DESIGNING THE USER INTERFACE

Perhaps the most important criteria in designing a user interface is that whatever method is chosen, it should be "user friendly". What user friendly actually means may be different in different situations or with users of varying experience with computer systems in general and with the individual program. However there are certain attributes of user friendly systems that can be generally accepted. Crosley (1985) discusses four aspects of user interface design. In the first instance, if the program is to be used by people with a variety of experience, the system directions or prompts should be concise, but clearly understandable by non-technically oriented personnel. Some systems allow the user to set the level or verbosity of prompts and responses to reflect the familiarity of the user with the system.

Second, the system should allow the experienced user to take short cuts where feasible. For example, repeatedly having to go through several menus to perform a task that the user is familiar with can become very tedious. Third, the system should be robust enough not to fail if the user makes a simple mistake, and " ... error messages should be clear in their meaning and provide some direction on how to correct the problem ..." (Crosley, 1985; 134).

Fourth, the user should be able to obtain "help" at any time, which may take the form of more detailed directions which more fully illustrate what is required of the user, or give the user a list of options available in the current situation, to perhaps offering some further explanation of why a particular question has been asked or how the system arrived at a certain decision (this capability of answering how and why is a particular feature of many expert systems).

Additionally, one point not considered by Crosley, it is desirable that the user can easily and quickly exit the system at almost any time without destroying files etc., and if possible return at a later date and quickly resume work without having to re specify large amounts of basic information.

One consideration that does not seem to be given much attention is the maintenance of records of what has been done, particularly in interactive sessions.

Several systems do maintain journal files so the situation can be recovered after a system 'crash', but these are seldom retained after a job has been completed. Diagnostic files may be produced, but these document every step undertaken. What is required is the ability to return and say "I made this map last year. Now I want a similar one showing ...". Using a large mainframe system, with good record keeping this should be possible, although it may be time consuming, requiring information to be restored from backup tapes. On smaller systems it is obviously not practical to record the details of every map produced, but an expert system should at least have retained the 'knowledge' to create such a similar map with as little input from the user as possible.

In the long term it should be expected that expert systems will communicate with the user in natural language, perhaps even by voice communication. Ideally the user would simply enter the type of map they want and the computer would interpret their request. This is a longer term aim however, and initially it would seem that as there is a specific goal to achieve in using an expert system, some form of structured interface largely based on the use of menus is more likely to be the main form of interface. The use of an expert system is quite different to that of a general purpose word processor or spreadsheet, therefor the use of the desktop metaphor is not the most suitable although this does not preclude the use of windows, the mouse, etc. when appropriate.

CHAPTER THREE

The Application of Expert Systems in Design, Cartographic Design and Mapping.

> Probably the user most at risk is the one who produces maps or other graphical output for his own use or for limited circulation. ... the user, in designing his output, will often use an interactive graphic facility and therefore he needs to optimise the information appearing on the screen appropriate to his particular expertise. ... [also] the final product may appear on a totally different medium, e.g. paper, which leads to further problems.[1]

EXPERT SYSTEMS FOR CARTOGRAPHY

A broad overview of possible applications of expert systems to "cartographic processes" is provided by Granklanoff (1985). He also tries to assess quantitatively the suitability of expert systems for various mapping tasks. Each of 17 mapping tasks from geodetic control to printing were assessed for their suitability by eight mapping experts using standard criteria. Although a very limited study, interestingly several tasks related to design feature near the top of the list for suitability, including generalisation and symbolisation, and feature selection and placement (Table 3.1).

Robinson and Jackson (1986) also identify a number of broad areas of cartography and digital mapping where expert systems could be of benefit. Their list includes: Manual and Automated Map Design; Digital Data-base/User Interface; Cartographic Education and Training; Spatial Data Error-train Analysis; Data Capture and Storage Standards; Data Format and Transfer Standards; and Replacing Cartographers. This is a very wide ranging list encompassing most areas of cartography, although they see the last entry on the list as being impractical for several reasons, not the least being the need for cartographers to provide their knowledge and monitor the achievements of automated systems.

A more recent review is provided by Buttenfield and Mark (1991). The purpose of their chapter is to present the design criteria for a cartographic expert system, essentially for map design. They note that the concept of a full cartographic expert system (CES) which could effectively design a wide range of maps over a wide range of scales from a single database is a monumental task. They review recent work on CES under three headings: generalisation,

[1] Robinson & Jackson, 1986; 431

symbolisation and production which they view as the logical major components of producing a map. Their table (Table 3.2) illustrates their view of the applicability of experts systems and progress in their development at that time. They consider this ability to split up cartography into a number of 'relatively easily isolated' sub tasks accounts for the relatively high volume of published articles on certain aspects of design (ibid.; 136) and hence makes cartographic design and production a suitable candidate for an expert systems approach. While this view is theoretically possible, in practice there is considerable interaction between various sub tasks, which probably accounts for many poor maps in the visual / aesthetic sense.

Table 3.1
Suitability of task for applying expert systems

Rank	Task name
1	Source Evaluation
2	Source Selection & Compilation Planning
3	Generalization and Symbolization
4	Feature Selection and Placement
5	Stereo Photogrammetric Plotting
6	Typesetting & Type Placement
7	Geodetic Control Identification
8	Color Separation Proofing
9	Overlay Proofing
10	Analytical Triangulation
11	Mensuration
12	Distribution & Shipping
13	Inventory & Stockage Control
14	Press Printing
15	Engraving (Scribing)
16	Plate Making
17	Negative Preparation

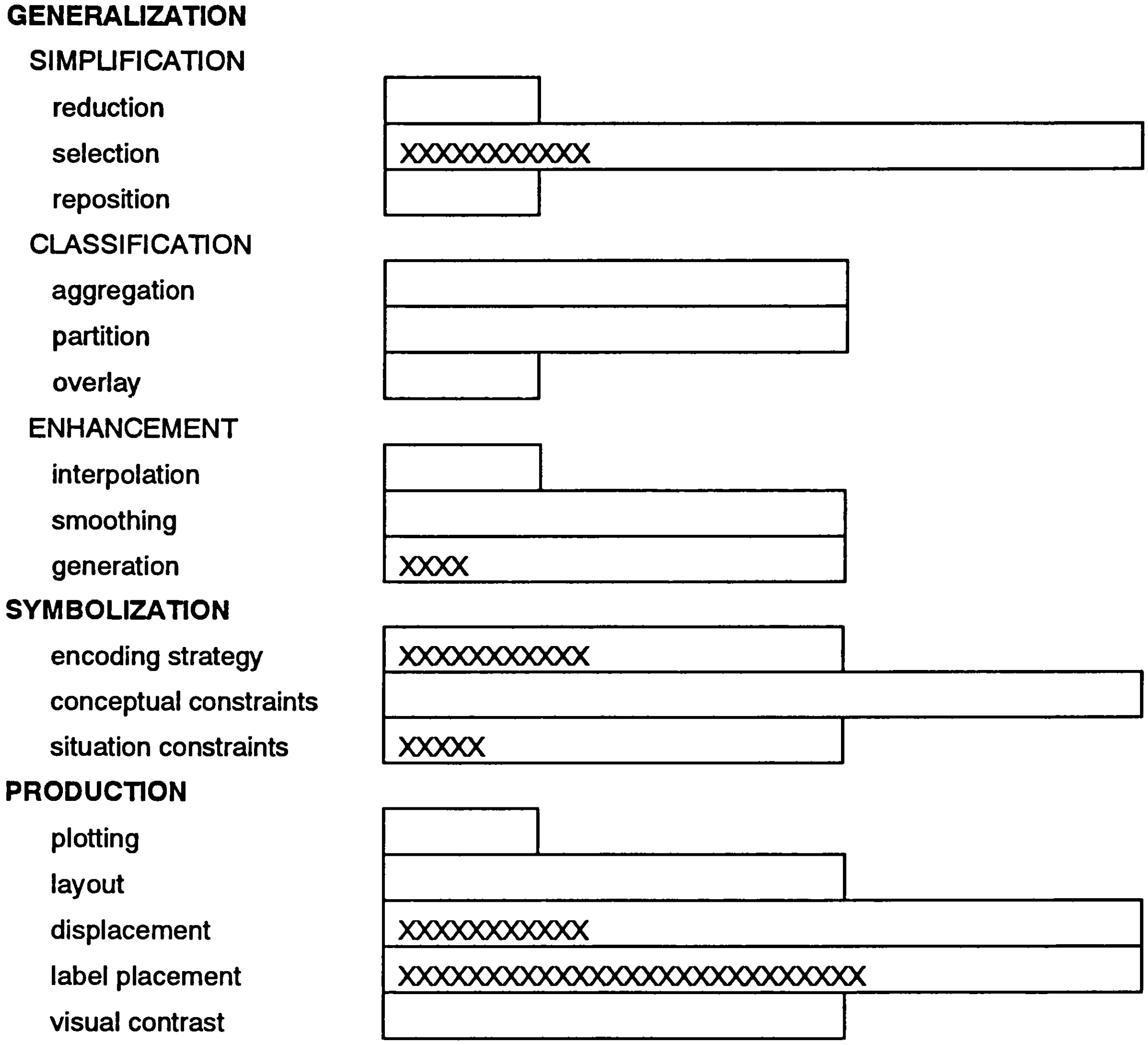
after Granklanoff (1985, p.621)

Further overviews of the application of expert systems in cartography are also to be found in Forrest (1991 & 1993 - see Appendix G), Mark & Buttenfield (1988) and Buttenfield & Mackaness (1991).

The emphasis of the remainder of this chapter is on reviewing systems falling into the first category on Robinson & Jackson's list and generally the middle class of Buttenfield & Mark's, but before concentrating on map design it is worthwhile reviewing some examples from the broader areas of design. Finally a brief summary of other applications of AI and ES to cartography and mapping is provided.

Table 3.2

Role of expert system in map design.



boxes represent potential application

XX = represent progress

after Buttenfield & Mark, 1991.

EXPERT SYSTEMS IN COMPUTER AIDED DESIGN

Computer aided design (CAD) systems have been available for some time, and are now extensively used in many engineering, architectural and similar applications. These systems rely upon human experts to solve design problems based upon experience, specialised knowledge and engineering judgement, using the computer only for analysis and drawing. Conventional CAD systems are based upon the assumption that feasible solutions exist to solve design problems, but this premise does not always apply in real-world design (Jansen & Puttgen, 1987). Human experts frequently have to deal with problems where all constraints and objectives of the design cannot be met. Thus, there has been considerable interest shown in the use of expert systems to enhance the capabilities of CAD systems (e.g. Begg, 1984, Gero, 1987, Tomiyama & ten Hagen, 1987 a, b).

'Design' Expert Systems.

The 'traditional' role of expert systems has been that of a diagnostic tool. Authors such as Waterman (1986) have provided detailed descriptions of a wide range of expert systems, but few of these are related to design or have a design component. Of those described in the pre 1990s literature as 'design' system, only DEC's XCON (see chapter 1) appears to have been in continual commercial use. However, systems such as XCON are perhaps more appropriately called 'selection by constraint' systems rather than approaching the general concept of 'design'. Although there have been several discussions about the use of AI in design for some years, it is only recently that ES technology has been examined for its general applicability to design.

Oxman and Gero identify two approaches to the application of expert systems in the design process: first, 'design synthesis' where the ES is a design generator; and secondly, 'design diagnosis' where the ES functions as a design critic to evaluate, criticise and recommend corrections to designs.

> In both modes of operation solutions are generated before they are analyzed and evaluated. . . . Both the way in which knowledge is represented and the way in which it is applied in the inference mechanism must in design application, recognise the recursive nature and multiple modes of design paradigms.
>
> (Oxman & Gero, 1987; 4)

They also state that a design which is generated according to rules should be assessed through models of performance, using some form of performance or evaluation knowledge. This additional knowledge is used to check the validity of a system's solution against its knowledge base of performance requirements. While one can generally see the applicability of this function, it is, however, unlikely that it can be rigorously applied to the design of maps which are notoriously difficult to evaluate in any quantitative manner. Any true evaluation of map design must go beyond assessing the spatial relationship between objects or the calculation of the amount of information.

Where design evaluation can be of considerable practical use is in systems such as PREDKIT, a system for the design (i.e. the layout of components) of kitchens (Oxman & Gero, 1987) where there are obvious and generally simple functional requirements, such as adequate ventilation and light. Despite its apparent simplicity, PREDKIT, which has under 100 rules of the object-attribute-value approach, shows that rule based systems can be utilised for simple design problems. Oxman and Gero do however state that from their experience simple rule-based knowledge bases are insufficient for more advanced modelling of design problems and intend supplementing PREDKIT with a frame based semantic modelling system.

Similarly, a system called ASDEP for designing power plant auxiliary electrical systems has clear guidelines for evaluating design quality, such as operational performance, reliability, maintainability, flexibility, expandability and cost. Interestingly, this system specifically aims to produce good or satisfactory designs rather than optimum ones (Jansen & Puttgen, 1987).

Yet again the basic functioning of this system is not compatible with the map design problem. ASDEP works by producing an initial design which conforms to the lowest cost and minimum reliability model which is then refined to produce a satisfactory design. An alternative strategy would be to start with a highly reliable but high cost design and modify it to achieve lower cost while still meeting stated constraints. The design evaluation in ASDEP is performed by two 'critics'. The first ascertains whether the physical constraints imposed on the design are satisfied, and the second evaluates overall design reliability.

ASDEP uses a simple rule structure and chains rules in the from IF <antecedent> THEN <action>, although it also incorporates meta-rules to direct the system. It also incorporates skeletal plans into its knowledge base. Contrary to most expert systems, ASDEP uses little or no backtracking, progressing in an orderly fashion from initial designs to detail designs.

As in many other fields the literature on design expert systems has expanded in recent years and there are now several books on the topic (e.g. Coyne et al., 1988; Gero, 1987; Pham, 1991; Roseman et al., 1988; ten Hagen & Tomiyama, 1987; Yoshikawa & Warman, 1987) and three conferences have been held entitled Artificial Intelligence in Design (in 1991, 92 & 94)[2]. A wide variety of application areas have been attempted including aircraft design (Morris, 1985), ship

[2] A paper describing the concept of the system developed here was submitted to the first of these conferences. Despite favourable comment from the referee and encouragement to publish the paper in the form submitted, the organisers were not interested a paper concentrating on cartography for the conference.

design (Akagi, 1991), building design (Newton, 1986), landscape design (Hsu, 1992), fixture design (Nee & Poo, 1991; Pham & de Sam Lazaro, 1991) and design for assembly of components (Gairola, 1986), although most reports are of experimental or theoretical systems. Unfortunately, due to the cost of creating sophisticated expert systems, details of working systems are rarely made freely available.

Similar to the trends in recent cartographic literature on expert systems, although to a much greater extent, considerable emphasis has been placed on formalising the design process in an attempt to derive a "knowledge-based model of design" and several conferences and books have been devoted to this theme (e.g. Coyne et al., 1988, Earnshaw, 1987, Yoshikawa & Warman, 1987). This is based upon a much larger body of literature on the design process than is available for cartographic design, and is a natural transition from previous attempts to explain design. Apart from communication theory and a few notable exceptions (e.g. Bertin, 1967; Keates 1982), cartographic design literature is generally lacking in this type of introspection. The aim of these studies is intelligent CAD (ICAD) systems which act as "intelligent design assistants" (Tomiyama & ten Hagen, 1987a, 1987b).

Interestingly, despite the increase in literature on Expert Systems for design in general, Architecture, a field which makes substantial use of computer aided design, appears to have been little influenced by expert systems. Searches of two architectural bibliographic databases carried out in October 1990 revealed only 27 and 7 references on 'expert systems', of which only 4 and 3 respectively related to 'design'.

It emerges from the literature that most design problems attempted with expert systems so far have neither the flexibility nor the constraints imposed on map design. These terms may seem contradictory: the flexibility of map design refers to the ability to select, simplify and combine information, as well as considerable flexibility in its symbolisation, whereas the constraints include the requirement of the geographical rigidity in location of most information and the need to maintain relationships between many varied types of information.

Thus, while some guidance on general principles may be gained from examination of expert systems in other fields of design, it seems unlikely that significant use can be made of their results in developing a cartographic design expert system. Parallels can however be seen. Several authors suggest that frames provide the most appropriate knowledge representation method for design (Cao et al, 1990; Coyne et al., 1988; Iudica, 1989; Landsdown, 1988), a conclusion reached independently for cartographic design both in the present study (see

Forrest 1992a and Chapter 7), by Wang (Wang, 1992) and by several other cartographic researchers. Also, in examining the automation of floor plan design, the aim of which is to "... create two-dimensional layouts based on topological, geometrical functional and aesthetic constraints", Coa et al., observe that "Because of the combinatorially explosive nature of the search problem it is impossible to search exhaustively for a solution" (1990; 213). Experience has shown that this applies equally to map design!

One interesting comparison with the general trend in design experts systems generally and their cartographic counterparts is the intended user. The concept of ICAD and the 'intelligent assistant' is clearly intended for trained designers, not untrained users of CAD systems, whereas most cartographic design expert systems seem to be aimed at the cartographically illiterate user. In this sense, cartography suffers along with other graphic arts such as typography where it seems anyone with access to a computer and a desk top publishing package is an instant typographer! The recent rash of kitchen designer and garden designer packages at low cost perhaps provides some parallel to the cartographic situation, but these are specifically intended for 'amateur' use, not professional or pseudo professional use. It is unlikely that plans for a home extension produced on one of the low cost CAD systems by a non qualified person would meet the approval of the Local Authority Building Control Officer, but no such checks exist on the use of cartographic products.

EXPERT SYSTEMS FOR COMPUTER AIDED CARTOGRAPHY

If the number of publications using the terms in their title is any indication, then in the last few years there has been considerable interest shown in the application of artificial intelligence and expert systems to cartography. Most are theoretical considerations of what can and might be done, or what can't, and while a few do report actual working examples, these are all of fairly limited scope or sophistication. A small number of publications try to deal with the general problem of map design, or significant portions of it, but more are concerned with specific aspects of map design and production, such as line simplification, name placement, or symbol selection.

It is fairly obvious that much of the research has been carried out by people who have little experience in the design and production of maps, nor indeed do they appear to have consulted acknowledged cartographic 'experts'. The tendency has been to rely on a relatively small portion of the cartographic literature, much of which is theoretical in nature. As Forrest and Pearson (1990) have shown, there is a large body of literature giving **practical** advice on map design, and more attention should be given to existing maps which solve the problems automation is attempting to solve.

Systems covering broad areas of design and symbolisation

Perhaps the first reported application of AI and ES to cartography was that of Poiker, Squirrel and Xie in 1982. Programs that may now be classified as being expert systems may have been in existence before then, but the terminology does not appear to have been used in the cartographic literature before this.

The proposed system was termed an 'Intelligent Cartographic System', as it combined current developments in computer assisted cartography and in artificial intelligence. The paper sets out a tentative framework for a system that could learn interactively as the cartographer creates maps, and would attempt to combine the artistic talents of the cartographer while freeing him from the repetitive work that computers can perform quickly and accurately. As the system relies on knowledgeable cartographic input it is clearly not intended for use by non-cartographers.

Poiker et al., recognised three areas in the production of digital maps: data input; geometric processing; and map design. These are connected by two interfaces: the data base and the display file. They briefly mention that some subset of information will be extracted from the database to form the display file, but do not expand upon this operation. Their view initially is that the cartographer will primarily be responsible for presenting the information, having been told what to select. This leaves a very narrow area of map design to be resolved, i.e. representation. Later they do address the problem of data selection and recognise that there must be interaction between data selection and design, although their view that selection, simplification and classification as aspects of generalisation are easily solved is rather naive.

What they in effect describe is a system for resolving spatial conflicts when features overlap due to the scale of the display. All features are initially placed using their correct co-ordinates. If two features collide (overlap) then one or both must be moved. The aim is to position all features with the minimum of displacement to all of them.

Solutions to these problems form the knowledge base. Conflicts are resolved by three processes: a) Using hierarchies of feature attributes, i.e. more important features or attributes take precedence. b) Using negotiation protocols in the form of rules such as 'IF a river and a road overlap THEN move the road. In this case move would be a function programmed into the system for these eventualities. c) By manual intervention, when neither a) nor b) can solve the conflict. The cartographer has the option at this stage either to solve the problem directly, or to teach the system a new rule to solve it.

This last option is very important. Initially the system would contain few rules, only a few of the most important and obvious hierarchies and protocols. The system will learn from the experience of the cartographer in resolving conflicts as they arise, although the functions actually performing the solutions would have to be programmed separately and explicitly.

As with many early writings on ES, this appears to have been a purely theoretical exercise, with little information given as to how such a system would have been implemented in practice. The authors were, however, quite narrow in their expectations of what expert systems might achieve, something not typical of many early studies.

Muller et al., (1986) developed what appears to be a fairly simple, but nevertheless operational cartographic expert system for deriving the specifications for mapping thematic information. However, it is not a complete system, lacking a fully developed user interface. Also, it does not actually produce maps, it " ... defines the most suitable graphic representation" (Muller et al., 1986; 568), so it is acknowledged as only covering a small part of traditional cartographic expertise.

Usefully, Muller et al., discuss at some length the background to the decision making process for executing map design, and find, like others, there are no universally accepted rules, although some convergence of general principles is apparent in the literature. They rightly point out that this type of uncertain, non-numerical knowledge is unsuited to traditional computing methods and that expert systems are well suited to represent, manipulate and modify cartographic knowledge.

A major emphasis of Muller et al's study was the development of a formal model of 'cartographic knowledge' and how to represent this. The model is in the form of a two level hierarchy of declarative knowledge for on the one hand the input (i.e. map requirements) and on the other the output (i.e. map specifications). There are nine categories of input, comprising 40 elements, and 55 output elements grouped into ten categories, as illustrated in Table 3.3. Generally, but not in every case, the output elements in each category are mutually exclusive.

The declarative knowledge is represented as a matrix of 40 by 55 cells with a value between -5 and +5 assigned to each cell, adopting the method used by Naylor (1983). This value indicates the relatedness of the input type to the output,

+5 indicating an imperative relationship, -5 an exclusive relationship, and 0 uncertainty or ignorance. Values between 0 and 5 represent degrees of positive relation and values between 0 and -5 degrees of non-relation.

While this approach is open ended, in that the matrix can be expanded, it would become extremely unwieldy if developed for the production of more complex maps. There is also the problem of depicting several phenomena with conflicting specifications on one map: no solutions are offered for such situations. This approach is well suited to more restricted problems such as the selection of an appropriate map projection. Such a scheme was developed in an early phase of the current project, but not pursued as it was peripheral to the main aims of the project, and the model was felt to be too simple to expand to the full range of problems encountered in map design.

The system described by Muller et al., makes decisions by simple arithmetic sums of the coefficients of the matrix for each output option corresponding to the active input elements. For each output group the element with the highest score is selected. If a tie occurs, all of the contenders are displayed, and presumably the user would select which option to adopt.

In the example cited by Muller, the rule matrix was formed by feeding 40 examples into the system ranging from topographic to statistical maps and from "large" to "small" scale. (Examples ranged from surveyed property lines to highway traffic across Canada.) The system, like many other proposed cartographic expert systems is claimed to be capable of designing any type of map at any scale. As the criteria for mapping various phenomena changes dramatically with scale, this must be seen as being an unrealistic objective of such a simple model. After the initial 40 examples were fed into the system to construct the knowledge base, the system was tested to try and regenerate the examples. Interestingly, none of the 40 examples were given the correct specifications, and only after considerable modification to the knowledge base (up to 24 iterations) was the system able to produce correct specifications for the forty examples used to create it. No testing of the system for producing maps other than those used to create the knowledge base is reported, a clear limitation.

Table 3.3

Muller's input and output elements.

INPUT CATEGORIES

DYNAMIC

1. Movement X-Y
2. Static X-Y
3. Time Series

DATA TYPE

4. Network
5. Volume Data Over Area
6. Volume Data Along Line
7. Volume Data At Point
8. Area Data
9. Line Data
10. Point Data
11. Spatially Discrete
12. Non-areally Related Census
13. Areally Related Census
14. Differentiable Surface
15. Continuous Surface
16. Spatially Ubiquitous

ACCURACY REQUIREMENT

17. Neighborliness
18. Angular Accuracy
19. Length Accuracy
20. Positional Accuracy
21. Area Accuracy

QUERY LEVEL

22. What is Related to What
23. Where is What
24. What is There

NUMBER OF COMPONENTS

25. 1 Component
26. 2 Components
27. >2 Components

IMAGE DIMENSION

28. 3-D
29. 2-D

MEASUREMENT

30. Ratio
31. Absolute
32. Ordinal
33. Categorical

MAP SCALE

34. Small Scale
35. Medium Scale
36. Large Scale

FUNCTION

37. Advertising
38. Communicating
39. Processing
40. Storage

OUTPUT CATEGORIES

PROJECTION ASPECT

1. Oblique
2. Polar
3. Transverse
4. Equatorial

PROJECTION SURFACE

1. Conical
2. Cylindrical
3. Azimuthal

PROJECTION QUALITY

1. Aphilactic
2. Equidistant
3. Equal Area
4. Conformal
5. Continuous
6. Interupted

PROJECTION VIEWING POINT

1. Vertical
2. Oblique

REPRESENTATION TYPE

1. Area Cartogram
2. Linear Cartogram
3. Graphic Chart
4. Color Patch Map
5. Simple Proportional Symbols
6. Compound Proportional Symbols
7. Line Map
8. Flow Map
9. Dot Map
10. Choropleth Map
11. Trend Surface
12. Isopleth
13. Isometric
14. Stereogram
15. Hypsometric Rendering
16. Hill Shading
17. Inclined Contours
18. Physiographic Diagram

NUMBER OF DISPLAYS

1. Map Series
2. One Map (Several Components)
3. One Map (One Component)

CHROMATIC

1. Polychromatic
2. Monochromatic

GRAPHIC PRIMITIVE

1. Form
2. Orientation
3. Texture
4. Color
5. Value
6. Size

GRAPHIC IMPOSITION

1. Volume Symbology
2. Area Symbology
3. Line Symbology
4. Point Symbology

GENERALIZATION

1. High Generalization
2. Low Generalization

After Muller, 1986, pp. 560-1

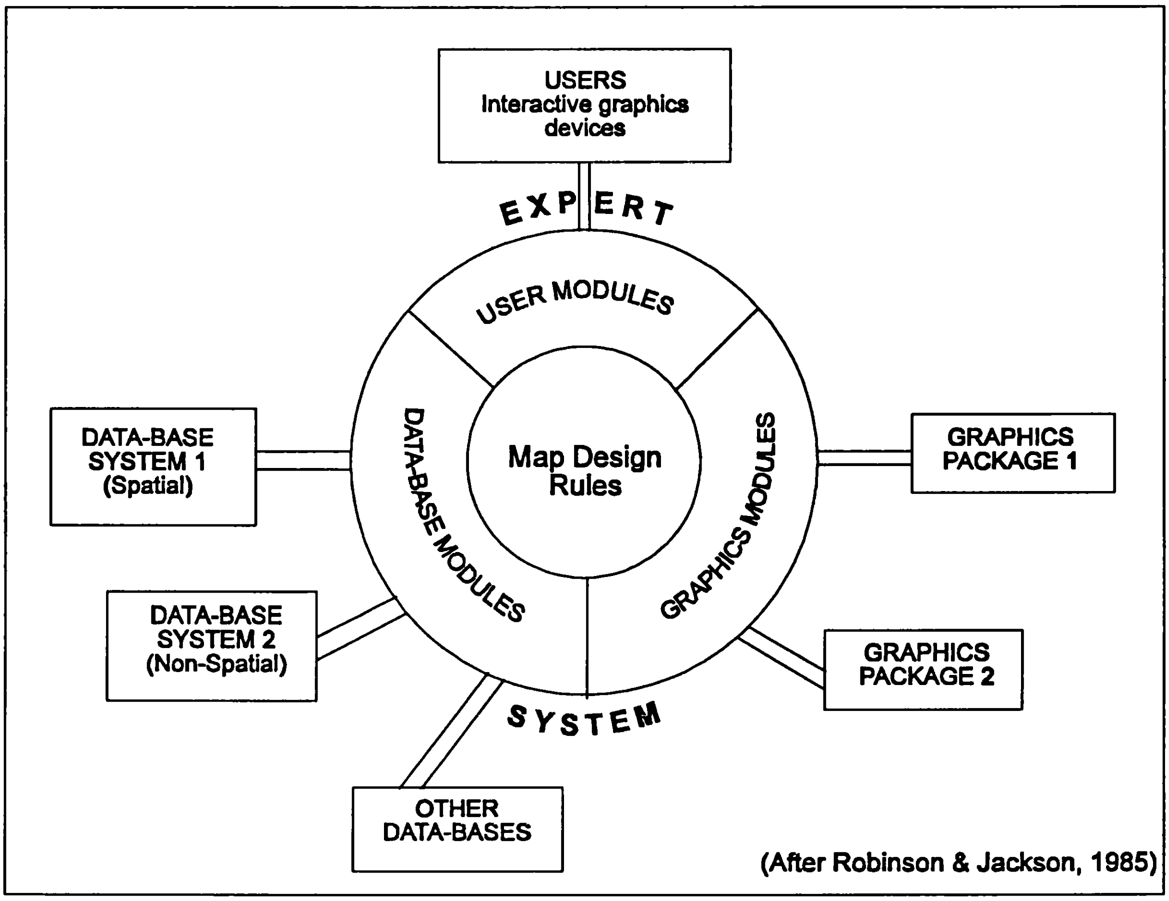
While this simple approach clearly does work, as more and more variables are considered the matrix would of necessity become very large and therefor development is limited. There is some utility in the model for certain aspects of the map design process, such as using the scores to represent the likelihood of including particular datasets in a range of map types.

MAP-AID. One of the larger projects on expert systems for map design reported to date is the MAP-AID project and its offshoots initiated in 1984 by the Thematic Information Service (TIS) of NERC in the U.K. (e.g. Robinson & Jackson, 1986: Mackaness et al, 1985). This project was initially set up as a co-operative venture between the TIS and collaborators in several U.K. universities and polytechnics, although the main project never reached fruition.

The proposed MAP-AID system (Figure 3.1) is composed of three elements: an expert system; a data base system which holds spatial and associated attribute data; and a graphics package (Robinson & Jackson, 1986; 435-7). The expert system itself is divided into four modules. The 'core' contains the map design rules and other information in the form of rules. The user module operates as the user friendly interface, translating user input into a format suitable for processing by the core, and vice versa. The data base module similarly translates between the external data base(s) and the core. The fourth and final part, the graphics module, translates standard core graphical output into the requirements of the particular graphics package implemented, thus allowing different devices, etc., to be used.

Surprisingly, no specific mention is made of an inference engine. One assumes this must be incorporated into the core along with the knowledge base. The structure of the knowledge base is not discussed either, although Jackson earlier (1984, pers com) mentions the use of the Naylor matrix model as used by Muller et al.

Robinson and Jackson point out that rule identification often leads one from a simple statement into a complex issue, for example " ... a simple sounding rule . . . such as 'don't use too many colours on complicated data' leads one into questions of perception of colour, measurement of colour, spatial interaction of colour etc." (1986; 437). For this particular example they list ten factors that should be considered and/or measured.



(After Robinson & Jackson, 1985)

Figure 3.1
The structure of the Map-Aid Expert System

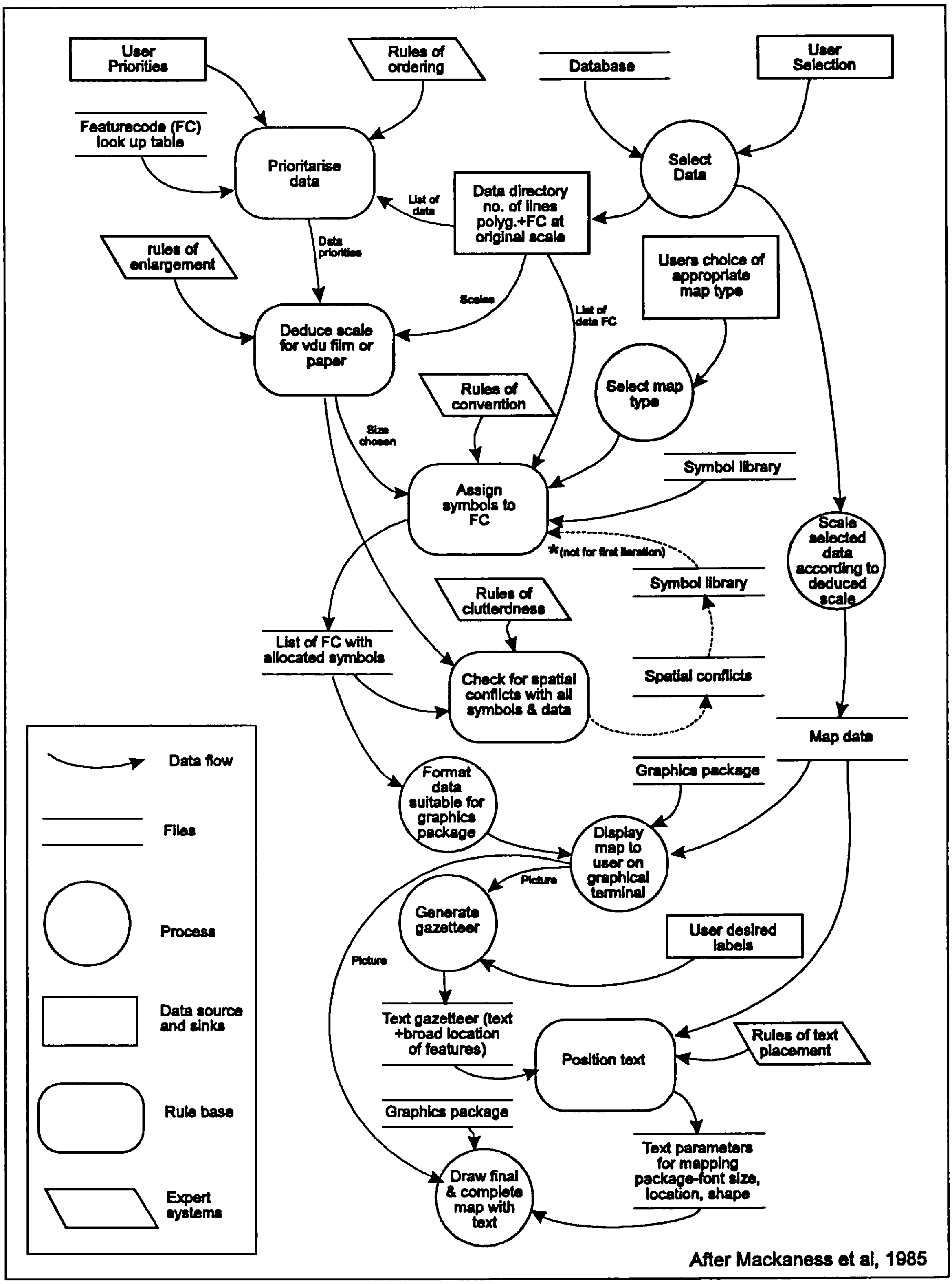
The MapAid concept is developed considerably further by Mackaness et al (1985), and although this was mainly theoretical, a prototype system was developed. Like others they recognise the large amount of literature published on map design in recent years and assume that this means that there is a considerable domain of expertise available. They do not, however, appear to have analyzed this literature for its applicability to rule formation in any depth. They foresaw the main problems of creating cartographic expert systems (CES) as being the representation of graphic design knowledge and hardware dependence of the final output (1985; 11). They also believe that a knowledge based system:

> ... must be capable of quantitatively assessing the spatial relationship between point, line and area data in a geographical data set whilst constraining its search procedure.
> (Mackaness et al, 1985; 11)

It must also be capable of associating measures of visual acceptability with partial design solutions. They do not define these terms nor offer any solutions to the problem of evaluating how good a design is. This lack of any systematic evaluation procedure for map design is a recurring problem with any attempt to automate design.

A major component of the study, like Muller's, is an attempt to model the map production process for automation, their model being illustrated in Figure 3.2. However, they make the common assumption, or over simplification, that one model can be used for all maps at any scale, and suggest that a system like MapAid could design Topographic, Oceanographic, Hydrographic, Geological, Political and Statistical maps, among others at scales from 1:500 to 1:20 000 000. It is extremely unlikely that, given the current state of development of cartographic expert systems, such a system is feasible, even if it were desirable, which is questionable. In many cases where fully developed designs have evolved, such as hydrographic charts, a specific system to aid in their production is likely to be more useful.

In the description of the MapAid system, the first stage is determining some of the map author's basic requirements, such as the type of map required, whether any data to be represented in a certain way, what conventions are appropriate, and whether there are any special needs of the user. They then immediately discuss the symbolisation of the data, despite quoting Monmonier's (1981) comment that the selection of content is equally if not more important than the symbology used in the design of the map. Mackaness et al., set out eleven sets of rules, (Table 3.4) the first four of which '... govern the choice [presumably of symbols] for each component' (1985; 16), and a further six dealing with the evaluation of conflicts that may exist between these symbols. The final set of rules deals not with map content, but with ancillary information such as titles, legends, etc. This procedure is similar to, although more comprehensive than that proposed by Poiker et al., (1982) with every item being symbolised ideally, then conflicts resolved.


User Priorities
Rules of ordering
Database
User Selection
Featurecode (FC) look up table
Prioritarise data
Select Data
List of data
Data directory no. of lines polyg.+FC at original scale
Data priorities
rules of enlargement
Users choice of appropriate map type
Scales
List of data FC
Deduce scale for vdu film or paper
Select map type
Rules of convention
Size chosen
Assign symbols to FC
Symbol library
Scale selected data according to deduced scale
*(not for first iteration)
Rules of clutterdness
Symbol library
List of FC with allocated symbols
Check for spatial conflicts with all symbols & data
Spatial conflicts
Map data
Data flow
Files
Format data suitable for graphics package
Graphics package
Display map to user on graphical terminal
Picture
Generate gazetteer
Process
User desired labels
Picture
Data source and sinks
Text gazetteer (text +broad location of features)
Position text
Rules of text placement
Rule base
Graphics package
Draw final & complete map with text
Text parameters for mapping package-font size, location, shape
Expert systems

After Mackaness et al, 1985

Figure 3.2
Data Flow Diagram of the Map Production Process

Table 3.4

Sets of rules governing the existence and placement of symbols

1.	Rules governing the choice of	Areas	
2.	"	Lines	
3.	"	Points	
4.	"	Text	
5.	Rules of conflict between	Areas and Lines	& vice versa
6.	"	Areas and Points	"
7.	"	Areas and Text	"
8.	"	Lines and Points	"
9.	"	Lines and Text	"
10.	"	Points and Text	"
11.	Rules of conflict between Locational and non-locational data and vice versa.		

After Mackaness, et al 1985 p.16

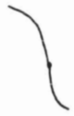
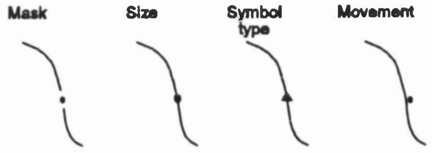
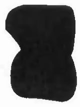
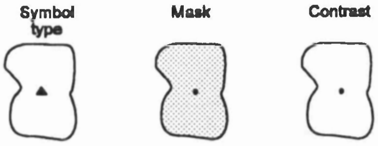
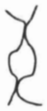
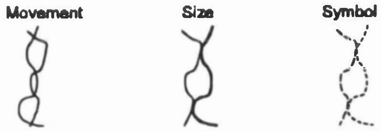
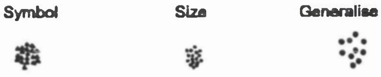
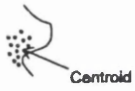
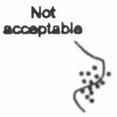
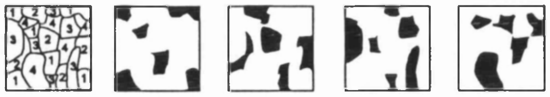
The whole philosophy of the MapAid project seems to revolve around symbols. "Once this [the map topic] was chosen the CES could select a particular field of symbols . . . along with their respective priority ratings (i.e. the relative importance of each symbol over one another)" (Mackaness et al, 1985; 18). This concentration on symbols rather than the representation of phenomena must result in limitations on the system. There are numerous cases where a phenomenon may be represented by a variety of different symbol types. For example, relief may be represented by continuous variation in tone (hill shading), area colours (hypsometric tints), lines (contours) or point symbols (spot heights) depending on the type of map and the other information to be represented. Thus, decisions about symbolisation depend partly on what other information is included on the map, cannot be made in isolation, and cannot always be standardised. Mackaness et al., view the selection of information for mapping as a 'process' (see Fig 3.2), not requiring rules and inference, although several CES for selection have since been developed (e.g. Richardson, 1989; Rusak Mazur & Castner, 1990).

The selection and priority of particular symbols are based upon a value between 0 and 5 assigned to each symbol for each of the possible map types they identify. If there is a conflict between two symbols and they have the same priority then rules will be used to resolve this (1985; 19), these rules being developed to a greater extent later by Mackaness (1986) and Mackaness et al., (1986). There is no discussion of what happens when two symbols of different priorities conflict, the implication being that the higher priority symbol takes precedence and the lower one is lost, but this is rarely a satisfactory solution.

While recognising that many areas of map design are complex, some problems are dismissed as being trivial, such as the " ... problem of knowing whether two colours are different is relatively simple ... " (1985; 21). Although this may be the case for colours in isolation, when colours are used for small areas of varying complex shapes scattered about a map, the perception of differences is less easy. It is not simply sufficient to say that two colours are different. Are they different enough? This involves complex questions of contrast, size of feature (or symbol), texture, etc. The creation of, for example, a graded series with significantly different, but not displeasing steps is not simple, although other researchers have examined some aspects of this problem for computer graphic displays (Dobson, 1983 & 1984, McGranahan, 1985 & 1986, Gilmartin, 1987).

The final visual appearance of the map is obviously of paramount importance. Mackaness et al., discuss the problem of 'clutteredness' and suggest that the problem might be solved by " ... better selection of colours for the range of data, changing the size and colour of the text, making symbols iconic, reducing the amount of data shown, or all of the above!" (1985; 22), but they do not offer any practical solutions at this stage to the determination of 'clutteredness' nor how to adjust map design to remove it. Some of their suggestions are of dubious value: iconic point symbols typically have to be larger than simple geometric ones, and even if the same size, it is difficult to see how this would reduce 'clutter'.

Later work by Mackaness (1986, & Mackaness et al., 1986) considerably develops the ideas of resolution of spatial conflicts when two symbols occupy the same map space, and also tackles the problem of evaluating map complexity. Figure 3.3 shows some of the evaluative functions performed and how some conflicts may be resolved. Some of the solutions are rather obvious and trivial, although they still require formalisation if they are to be incorporated into an expert system. Of particular importance and utility is the work by Mackaness on the problem of clustered point symbols and how they should be generalised or displaced to avoid overlap. His solutions depend upon cluster analysis and appear to provide satisfactory solutions even in difficult situations such as when in proximity to lines. These developments should obviously be included in any comprehensive cartographic expert system.

	FUNCTION REQUIRED	REASON FOR INVESTIGATION	POTENTIAL PROBLEM	POSSIBLE SOLUTIONS
1)	is a point on a line?	Points must not be obscured by lines		
2)	Is a point in a polygon?	Points must not be obscured by polyon symbol		
3)	For any two lines, how much segment overlap is there?	Line must not be obscured by line		
4)	For any points, are they clustered?	Points must not obscure one another		
5)	Total number of points in cluster?	If large number then spatial separation method must not be used		
6)	Centroid of the cluster?	To determine mean proximity of potential interference from lines and areas		
7)	With area of polygon and number of occurances, calculate free map area (FMA)	FMA used to evaluate complexity of maps, and use of symbols		
8)	Number of different fcs for lines	To determine FMA for lines		
9)	Number of different fca fo points	To determine FMA for points		

Feature	1	2	3	4	
Free Map Area	24	30	20	26	100%
No. of Occurances	6	5	6	5	22
Mean Feature Size (FMA / Occ.)	4.0	6.0	3.3	5.2	

After Mackaness & Fisher, 1987

Figure 3.3
Evaluation Functions, Problems and Solutions

The other main concern of Mackaness is the measurement of map complexity. The concept of 'Free Map Area' as described by Yoeli (1972) for areas, lines and points is used - see figure 3.3, items 7, 8 & 9. The 'Free Map Area' for a feature is the amount of space on the map not occupied by another symbol of the same type, i.e. the total map area minus the area taken up by symbols for that class.

> Free map area can be calculated for individual symbols, single data types, particular tones, or as an indication of the visual content for the map as a whole.
>
> (Mackaness, 1986; 20)

This can be calculated at any stage in the design process, and if certain thresholds are exceeded then the expert system will invoke rules to alter the value, such as reducing the size of symbols, reducing the amount of information shown, or re-classifying the data. As has been stated elsewhere, the evaluation of map complexity is not a simple problem and it is unlikely that such a physical evaluation yields particularly useful results, as perceived complexity by the map user is more important. It may, however, have some limited application in keeping simple maps simple, although in many designs the 'spaces' are important components, not just empty areas, i.e. white or 'unsymbolised' areas may be part of the overall classification, even if only representing a very broad general class. Also, in many cases it is the pattern of symbols that is important, not the physical space which they occupy. In such a case some correlation coefficient (e.g. see Unwin 1981) is likely to be as good a measure of complexity.

Inevitably in such a new field of study, rapidly growing in popularity and attention there will be several individuals or groups striving to form the appropriate methodologies. Mid 1990 saw the publication of several reports on systems for map design, including the background structure of the system described here (Forrest 1990), largely as a result of the 4th International Symposium for Spatial Data Handling (SDH90). Indeed, there appears to be a considerable convergence on the structure of cartographic design expert systems, although the concentration is on defining the relationship between information and representation. Several of the systems described, although appearing to be of a general nature, concentrate largely upon the symbolisation aspects.

The most extensive published information about a cartographic design expert system relates to the system developed by Wang (Wang, 1990; Muller & Wang, 1990; Wang & Brown, 1991; Wang 1992; Wang, 1993). The first of these articles concentrates on the handling of 'quantitative' information, in particular that relating to population and socio-economics, and develops a conceptual graph of the information and the relationships between various categories. The rationale is that this formalisation is required to establish the correct representation for the data. The knowledge is represented in the form of a semantic network showing concepts (entity, action or state) linked by conceptual relations. The purpose of this is to allow additional knowledge about the information to be added to a GIS, with the schema of relationships limiting the user from requesting unmeaningful relationships to be represented by the system, or limiting the representations appropriate in a given situation.

In contrast, Muller and Wang (1990) concentrate upon the representational aspects of map design, describing a system for " ... symbolic representation of statistical information based on physical or administrative units" (1990; 26). This is largely based upon Bertin's graphic semiology and is more comprehensively discussed by Wang (1992).

They identify seven types of statistical "maps" as possible representations, although this includes qualitative maps, which in the strictest sense do not represent statistical information (statistic: tabulated **numerical** facts - Chamber's Etymological English Dictionary). The representation of numerical data is limited to proportional symbol and choropleth maps, eliminating both dot maps and isopleth maps from consideration in order to " ... simplify the situation . . . [and limiting] the correspondence between data type and the map type to a one-to-one relationship" (Muller and Wang, 1990; 27). As will be discussed in Chapter 5, this is a very limited view of the relationship between geographic phenomena and cartographic representation, although clearly making the development of a prototype system much more achievable.

For each of the seven map types they list a number of possible "solutions" (i.e. set of symbols) based upon perceptual properties of the visual variables. The most appropriate solution is selected based upon several factors, including map use requirements, numerical analysis and conventional associations (Muller and Wang, 1990; 28). In some cases this is determined directly by the system, but more likely the user will be prompted for more information, usually by selecting options from menus. If more than one factor is involved in selecting the "solution", certainty factors are used to determine the most appropriate solution, although how these are applied is not fully detailed.

Once the map type and optimal solution have been determined the actual symbols will be assigned. To assist with this the knowledge base includes some conventions, e.g. woodland should be coloured green. If no conventions exist then colour and/or shape are assigned in sequence from lists. Size of point symbols is selected such that between 5% and 13% of the total map area is covered by the symbols. No details are given of how the sizes of proportional point symbols are then determined, nor of how lightness values are assigned to choropleth classes. Unfortunately, despite the obvious merits of the system, the article describing it is weak in both use of terminology and in explaining the many aspects of the working of the system, although some of the latter is better described in Wang's dissertation (1992).

A more comprehensive description of the ESSYD system is provided by Wang (1992). Like the present study, a frame based approach is used for the system and Turbo Prolog (in conjunction with Turbo C) used for programming, although a predominantly backward chaining procedure is followed. A hypothesis for Map Type (representation method) is set up and the inference engine attempts to establish evidence for this being correct. The system also incorporates probabilities and certainty factors to allow 'best first' search of alternatives, a further description of this aspect being given by Wang in 1993. In most situations the systems searches the knowledge base for all possible solutions, ranks these and presents the user with a list of possibilities with associated probabilities of being the preferred choice. The user then selects from the list, the system adopts this choice and proceeds to the next stage. No description is given of backtracking should the system fail at some subsequent stage.

The ESSYD system has built into it a module for selecting appropriate colours based on the ITC Ostwald colour chart (Wang & Brown, 1991). This module will select a set of colours for various area representation types. The system also includes mechanisms to explain how an answer was derived and why a question is being asked, although these answers are programmed into the system rather than being derived automatically, i.e. they are predetermined, not deductive.

Despite the obvious strong points of this system and its well developed use of certainty factors and their combination, the system still appears to be limited to determining the representation method and symbols for a single 'statistical' data set.

Jaakkola, Sarjakoski, Blom & Laurema (1990) similarly discuss the development of an expert system based upon a taxonomy of thematic maps. They consider there to be five phases in making thematic maps, summarised below:

1. Processing of numeric theme data.
2. Selection of presentation type [i.e. form of graphic representation].
3. Selection of base map.
4. Selection of the graphic presentation symbols and formation of the semantic relations between each symbol and the data it represents.
5. Selection of layout. [In effect, they describe compilation.]

The emphasis of the report is on the selection of the most appropriate 'map type'. In order to arrive at this Jaakkola et al., develop taxonomies of data, of queries and of map types. The tree structures of these taxonomies are well presented but, unfortunately, as with several other studies, the importance of the relationship between phenomena, information and representation is not discussed (see Chapter 5). Also some of the branches are not fully expanded, with the concentration being on the handling of areal data. The taxonomy of map types (i.e. representation methods) includes "raster" as one class of area maps, but raster is a specific data structure with the representation appearing as an artefact of this structure. It is possible to have, for example, a raster choropleth map or a raster chorochromatic map which have quite different characteristics in terms of cartographic representation.

These taxonomies are used to develop a selection table for representation method which is coded in the form of IF-THEN rules. For example: "IF type of data is A AND the point of view of the query is B THEN use map type C" (Jaakkola, et al., 1990; 716). In fact, although these rules have been developed, they were only used to create a series of maps to be stored as part of an electronic atlas, with several maps being created and stored for each variable and for combinations of variables. Thus, the user does not interact with the knowledge base to create his own map. Given the user's query, the system will display the most appropriate map from its archive, clearly greatly limiting the use of the knowledge base.

De Jong and van der Wel (1990) discuss the development of a cartographic expert system (CES) front end to a GIS for use by planners. They outline five stages in composing a map:

1. Data-retrieval from user-database and topological database and (re)classification of variables.
2. Choice of mapscale or mapsize and of layout-type.
3. Choice of map-symbols.
4. Key-generation.
5. Map-composition.

Much of the paper concentrates upon data quality and accuracy. In the verbal presentation of the paper it was clear that the concentration was again on map symbol generation, rather than the broader issues of design, and that item 2 above was only at a primitive stage. The selection of symbols is based on a multi-criteria analysis of weights based upon the relation between map purpose and type of data (similar to the Muller's CES). These weights were determined by a panel of professional cartographers. If more than one possibility remains, the user must choose which to use (de Jong & van der Wel, 1990; 725).

In the last of the reports on ES at SDH90, Kottenstein discussed the development of a knowledge based "symbol reference system." The aim of this system is " ... to organize the logical reference between the chosen data and the symbols representing them in the map" (Kottenstein, 1990; 776). Interestingly the "geometrical, substantial and graphic criterias [sic] of all data" are described by the user separately from the symbolisation stage, and this information is held as part of the database. It would appear, however, that the user is given little help in performing this task, nor in selecting the appropriate information to include in a given map.

In creating a map, the parameters loaded from the database are used as default values. These are then checked by the rules in the knowledge base. If this checking procedure produces a different outcome the default value is replaced. These new values are then checked against the knowledge base, this being repeated until the input and output values match, i.e. an interactive approach is used.

The development of cartographic design expert systems has not been limited to western institutes. Several Chinese researchers are known to be active in expert systems for cartography and GIS. The MAPKEY system is a knowledge based system for producing thematic maps (Zang et al., 1991; Su, et al., 1993). Like several others, including the system described in later chapters, a frame based approach is used to store structured knowledge about maps. The system allows data analysis, data classification, symbol design, colour assignment, map lettering and legend design, although it is not clear from published information how much of this is automated. Certainly photographs of maps said to be produced by the system, although relatively simple, do show acceptable results (Fig 3.4).

The later paper on MAPKEY (Su et al., 1993) again concentrates on structural issues and does not give details of which processes are implemented with expert systems. The list of 'map types' produced by the system includes "Choropleth, Dot, Proportional symbol, Isarithmic and Value-by-area", so that the emphasis would appear to be similar to that of Wang's ESSYD in selecting representation method and symbols for a single statistical theme.

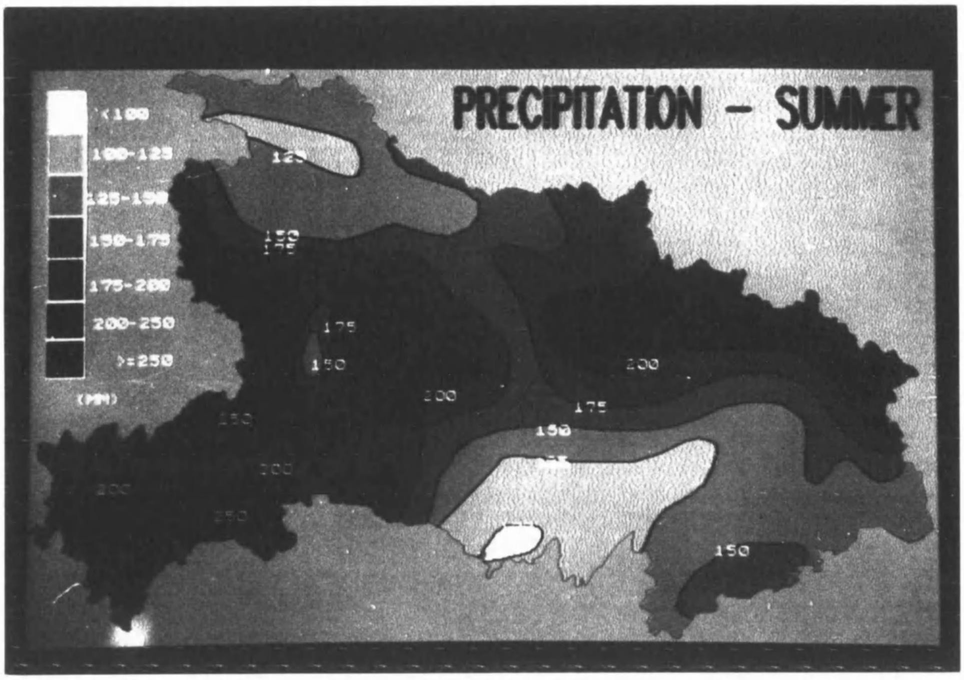


Figure 3.4
Example output from the MAPKEY system

Another flurry of activity in reporting the development of Cartographic expert systems may be found in the proceedings of the 16th ICA conference of 1993 including reports by Wang (1993) and Su et al (1993) mentioned above. No less than 13 papers are related to expert systems or AI applications, ten of which relate to design issues. Of these, 6 are reports from Chinese research groups, although several of these papers are very brief and superficial. Lu (1993) and Hua & Gao (1993) both outline the basic structure of statistical mapping expert systems and both propose the use a frame based structure.

Zhen et al (1993) describe a prototype system for nautical chart 'design'. They list six more or less independent subsystems comprising the system:

Sheet line determination
Mathematical factors determination
Compilation materials analysis and selection
Chart arrangement
Content element selection
Element presentation selection

Each of these have their own inference engine and knowledge base. Although introduced as a general system, the emphasis appears to be on the analysis of compilation material and determining the suitability of this, its up-to-dateness and accuracy for a given chart at a particular scale.

The system uses a combination of frames and production rules to represent knowledge, an example frame being shown in Figure 3.5. The inference engine is not described in detail, but uses both forward and backward chaining. Currently no certainty factors are included in the system which seems surprising given its emphasis.

Frame:	**chart2037**		
	slot:	name:	kind:
		value:	**common nautical**
	slot:	name:	usage:
		value:	**navigation**
	slot:	name:	scale:
		value:	
		if_needed:	**get_value (2037, scale)**
	slot:	name:	projection:
		value:	
		default:	**Mercator**

after Zhen et al, 1993; 99

Figure 3.5
An example frame from Nautical Chart Design Expert System

Despite the number of publications in the early 1990s reporting on the development of general map design systems, few appear to go beyond the function specification stage, and examples of maps produced by prototype systems are extremely rare[3]. This may indicate a lack of success in putting the theory into practice in an overall map design system and therefor the concentration of effort has moved to narrower problems for the time being. It could also be that, as Buttenfield & Mark have pointed out, "The actual development of a full cartographic expert system will require the work of cartographers, programmers, and knowledge engineers, over a period of several to many years" (1991; 146) and that none have yet reached the stage of being implemented in a fully working or commercial system.

[3] Even the dissertation by Wang (1992) fails to include examples of maps actually produced by the system reported on, and none of the other publications on this system include example map output.

An alternative approach to developing automated mapping systems has been suggested by Hutzler and Speiss (1993). In their 'knowledge-based thematic mapping system - the other way round' they are sceptical about true 'expert systems' but do see a role for 'knowledge based systems'. For a system to be considered a true expert system they believe all a user should have to do is load a data set and answer a few simple questions then be given the required result. Hutzler and Speiss consider this to be impractical currently but suggest that knowledge should gradually be built into existing map production systems to take care of, or assist with certain tasks. This is clearly a logical approach and is likely to be the model adopted by many commercial systems. It is clear, however, that Hutzler and Speiss are discussing systems for use by cartographers, not systems for users with little cartographic experience.

Expert systems for specific aspects of Computer Aided Cartography

Map Projections

One obvious application of expert systems in cartography is in the selection of map projections. This problem falls within the capabilities of diagnostic or classification systems. A very thorough report of the development of one such system is given in Nyerges and Jankowski (1989) and Jankowski and Nyerges (1989). Unlike many others, they commence with a detailed examination of the knowledge required to develop an expert system by examining the major works on the classification and selection of projections. From this they develop a conceptual structure for the knowledge base tying in both classification attributes and selection criteria.

They present a decision tree for map projection selection (1989; 36), although it must be said that there are a great many gaps in this, for example the branches for world equidistant projections are either 'centre at pole' or 'centre at city': Clearly this should only apply to equidistant azimuthal projections, not all equidistant projections. None-the-less, their methodology is clear and well illustrates the first step in developing an expert system.

This conceptual structure is later expanded upon and used to assess appropriate knowledge representation methods for MaPKBS, a map projection knowledge-based system (Jankowski and Nyerges, 1989). They investigated the use of frames and production rules for the representation of information in the knowledge base, and adopted a compromise solution allowing both unambiguous, innumerable information and ambiguous noninnumerable information to be used.

At the time of reporting the system was in the initial prototype stage using a proprietary expert system shell, but no results or analysis are presented.

Name arrangement.

A popular area of cartographic research in recent years has been the automation of name placement, particularly for those names relating to point features (e.g. Basoglu, 1982, Hirsch, 1982, Ahn and Freeman, 1983). That this is so is not surprising given the large amount of time reportedly devoted to this process in more detailed maps. Clearly, however, the proportion of time will depend upon the number of names and the distribution of features, but it is frequently significant. Most of the earlier attempts use an algorithmic approach and conventional programming, although some, such as the work by Hirsch (1980), do not guarantee a solution in all cases. Several of the more recent attempts at solving this problem have used expert system techniques to resolve conflicts in placement. Most systems appear to use the rules outlined by Imhof (1975) and Yoeli (1975), and concentrate on naming point features, not the more difficult area and linear feature names.

The first widely reported 'expert system' for name placement is the Autonap system by Freeman and Ahn (1984). This system uses heuristic knowledge about name placement based upon traditional cartographic conventions and procedures.

> The heuristic knowledge is embedded in a set of explicit rules that form the knowledge base of the system. The knowledge base is organised in such a way that it is relatively easy to add new rules or modify existing rules, or replace it altogether with a different knowledge base.
>
> (Freeman and Ahn, 1984; 544.)

Thus, the system shows several of the basic characteristics of a true expert system. The knowledge base is composed of twenty five rules, covering general principles, plus more specific rules for area, line and point feature name positioning. Freeman and Ahn recognise that "No single set of placement rules can satisfy all cartographic needs" (1984; 549). The rule based approach makes customising the system easier than the more traditional algorithmic approach. The system uses a graph search technique with the goal state being all names placed. If a name cannot be placed the system backtracks, removing names already placed until there is space for the unplaced name. Removed names are then replaced in new positions and the process repeated until all names are placed.

ACES (A Cartographic Expert System) appears essentially similar to Autonap. ACES uses a "procedural knowledge base" (Pfefferkorn et al., 1985; 400), presumably in the form of rules. Inference is carried out in a similar manner to

Autonap using an heuristic graph search method. It uses a decision tree to contain a history of previous strategies and where to look for possible solutions.

A rule-based system for name placement (NAMEX) written in Prolog has been developed by Cook and Jones (Jones, 1989, Jones & Cook, 1989, Jones, 1990, Cook & Jones, 1990). This was specifically designed for placing names based on the O.S. Routemaster database, but the system has more general application. For example, it has been used for automatic placing of crater names on lunar maps (Cook & Jones, 1990).

It must be pointed out that there are alternative views. Zoraster (1986) for example believes that 'optimisation techniques' are more appropriate to automating the name placement problem than the use of AI, and that optimisation techniques may even extend to the more general problems of selection of features for display.

One limitation of most name placement systems is that they do not interact with name selection, as would be done by an experienced cartographer. Kadmon (1972) described a database system to assist in the selection of names relating to settlements and this would provide a useful basis for integrating automated name selection and placement. Such an approach has been described by Langran and Poiker (1986).

Generalisation.

Another topic which has attracted the use of expert systems is generalisation. In one of the earlier published descriptions of a cartographic expert system, Nickerson and Freeman (1986) developed a rule-based system called MAPEX which uses English-like rules to describe the generalisation processes that may be used and the conditions under which they are invoked. This system goes considerably beyond the problem of line simplification, often seen to be the only problem of automated generalisation, although it is restricted primarily to linear features (including area outlines). Nickerson and Freeman recognise that the reason for generalisation is the reduction of information that can be shown as scale is reduced, the scale factor of change being of primary importance in most operations. The first stage in MAPEX is to generate an 'intermediate' map in which each symbol is replaced by one whose width is representative of the symbol at final map scale. Depending upon the scale change and the amount of detail, the system determines feature combination, deletion and simplification. The system will detect interference of features and displace them if necessary, propagating this to adjacent features where required (Nickerson and Freeman, 1986; 540-542).

The MAPEX system separates the rules governing generalisation from the mathematical algorithms required to carry out the geometrical changes. This has been found to be very beneficial and "the rules can be maintained and expanded independent from the rest of the automated generalization process" (Nickerson and Freeman, 1986; 555). In addition to global parameters the system has specific rules about different types of feature, such as roads, railways, hydrography, etc., and their interaction. Clearly if expert systems are to be of use to cartography this is the type of approach that must be taken, although it is likely that the rules will vary from system to system or even within a system for dealing with different scales or map types to meet requirements of different agencies, users, etc.

A useful approach is adopted by Richardson (1988) who used a relational database for rule-based feature selection of hydrography on small scale maps. As she rightly points out, in the initial stages of map design it is the overall information content of the map that is important, not the density of features or their representation (1988; 165).

The system described uses a large amount of attribute and relationship information stored in tables of a powerful relational database. This includes river segment length, discharge, distance to settlements, settlement size, etc. The user can enter queries to the system with several criteria to be met, and the resulting river network will be plotted. Although the facts stored are useful in extracting the appropriate river segments, the 'rules' would appear to be entered at run time, and thus little help is given to the user as to what usefully may be included in a map at a particular scale for a specific purpose, and the knowledge is not captured for future use.

The strength of this work is the appreciation that feature selection cannot be reduced to a simple quantitative rule, such as Topfer and Pillewizer's (1966) radical law. It also shows how the relational database of a geographic information system can be used to great advantage in map design if properly structured and applied.

Although most studies on generalisation concentrate upon line simplification, other aspects have received some attention. Robinson & Zaltash (1989) reported on the development of a system for the generalisation of buildings on O.S. large scale plans. The system (OSGEN) uses the Leonardo expert system shell to provide advice on the simplification, aggregation or removal of buildings when scale is reduced, although the range of possible reduction is currently limited to derived maps at 1:10,000 and 1:50,000 scales. Further development in this area was reported by Lee and Robinson (1993), although the description was still very much limited to the development of a prototype system.

Muller (1990) notes that there are two aspects to automated generalisation. The first involves the cartographic knowledge to assess the information requirements of the map and which generalisation processes to apply, and the second is the execution of these procedures. Muller concentrates on the first of these and examines the potentials and difficulties of a rule based approach, but no attempt is made to produce an operational system. Unusually, the basis of the knowledge was gathered by examining a series of topographic maps of the same area at scales between 1:1000 and 1:500,000.

A verbal description of the generalisations observed at the various scales is given, followed by a more formal representation of these statements as facts and rules using predicate calculus. For some reason the example given is for Imhof's table of the relationship between settlement population, scale and graphic representation for scales between 1:200,000 and 1:30,000,000, which does not seem to relate to the scale range considered by the rest of the study! Other possible knowledge representations such as a matrix similar to that use earlier by Muller et al (1986) and semantic nets are discussed briefly.

It is noted that generalisation at the larger scales was dominated by geometric processes such as simplification, enlargement and displacement, whereas at the smaller scales conceptual processes such as selection and classification were more evident (Muller, 1990; 329). One of the impediments to automated generalisation is what Muller terms 'catastrophic events' such as the change in symbolising a road by two lines representing its edges to a single line symbol.

A list summarising the sequence of generalisation processes involved is given, but further explanation of how these would be implemented is required. For example, item 1 states "Elimination of all objects which belong to categories of objects not to be represented on the derived map." Who decides what is to be represented on the derived map? It is hard to see how this could be done other than by manual intervention. The system cannot decide unless further rules were previously entered. Often it is not simply a matter of eliminating classes, but the basis of classes changes with scale. Muller gives two technical impediments to rule-based generalisation. These are that many rules indicate what **not** to do, but not what to do and that when rules do indicate what to do they do not indicate how to do it. While this may be true if one examines the problem superficially, a deeper understanding of the process should lead to the more specific rules that apply in specific cases.

On a technical aspect, Muller defines search strategies such as depth-first search and backward chaining as being heuristic (1990; 319). Backward chaining is not inherently heuristic, only being made so by using appropriate evaluation functions, and depth-first searching by definition is not heuristic.

Although Muller does point out that priorities vary in different countries, it would be interesting to see how the rules developed from examining a series of maps of an area south west of Hannover would apply elsewhere.

The most significant work on automating generalisation is without doubt Buttenfield and MacMasters' book on the topic published in 1991, resulting from a meeting held under the U.S. National Centre for Geographic Information and Analysis (NCGIA) initiative. This looks broadly at the issues involved in generalisation and although very few practical examples are given of operational systems, and no production systems are evident, the book is a worthwhile source of potential developments, research topics and potential solutions. A comprehensive review of the book by this investigator published in the Cartographic Journal (Forrest, 1993) can be found in Appendix G.

Generalisation is clearly an area of increasing research interest, with many GIS researchers active in the subject and many theoretical papers being published linking generalisation and AI (e.g. Muller et al, 1993; Wang, Wu & Wu, 1993). Often the term 'multiple representations' is used in preference to generalisation e.g. Buttenfield (1993), Kidner & Jones (1994). Arguably multiple representations (i.e. multiple scale dependent data sets and/or symbols stored in a system) and generalisation (i.e. deriving representations for smaller scales from larger scale data) are different, but the links are obvious and the intended end result - the use of appropriate representation depending on scale - is the same and often AI methods are incorporated (e.g. Kilpelainen & Sarjakoski, 1993).

Colour selection.

A final area of cartographic design relevant here that expert systems have been applied to is in the selection of colours for graphic display devices.

Samson and Poiker (1985) describe a rule-based system for selecting area colours for nominally coded (i.e. feature coded) polygon maps on a micro-computer display based upon principles suggested by Imhoff (1982). The system uses dither patterns on a basic eight colour system to produce 48 usable colours. This simple system uses rules such as 'large areas should not be too vivid'. It is an interactive system. Areas with a water feature code are identified first and assigned a blue colour. Once features are assigned a colour, that colour becomes 'reserved' and additional features are assigned colours to avoid clashing with those already assigned. The system allows for colour themes to be selected by the user, restricting the possible colour choices, but maintaining distinct differences. The selection of colour should be based upon the principle of showing nominal differences by hues and similarity of importance (equivalence) by similar lightness, which could be automated if a hierarchical feature coding system was used.

The system as described does not appear to include any backtracking which would allow previous selections to be modified as the situation develops. this lack of backtracking is clearly a great limitation and not a characteristic of a true expert system. This limitation could create problems if the total number of colours required is not known before allocation commences and there is a large number of classes.

A more extensive system 'CANVAS' has been developed for automatically assigning colours to classified satellite images and thematic maps (Trigg & Gill, 1988, Gill & Trigg, 1988). The reports point out many of the problems of developing such a system, such as the variability of phosphors on CRT displays, effects of incident light on the display, etc. Although the methods employed for assisting the user have wide application, the system itself is 'hard-wired' for the I^{2}S System 600 image processing system and therefore is not generally applicable to other environments.

As part of the development of a more general map design expert system (reported above) Wang and Brown (1991) and Wang (1992) also describe in some detail how area colour symbols may be chosen based on the use of the ITC colour chart. This system will select a set of colours for either a choropleth representation, or categorically defined (chorochromatic) areas.

Liang et al (1993) describe the development of an expert system for 'automatic colour design'. That this was a four year government funded collaborative research programme with relatively restricted goals gives some indication the magnitude of the task of developing cartographic expert systems. The main aim was to develop a system to match printed colours to screen colours, that is, the conversion from RGB colour specification to CMY or CMYK. The system appears to be well thought out and works interactively, with the user specifying broadly what is required in terms of colour scheme and the system automatically producing an appropriate colour palette.

Grossler (1993) reports on empirical research on the use of colour on population change maps and describes how such research could be used to develop a knowledge base for a cartographic expert system. An important aspect of this work is the recognition that while it may be difficult to extract knowledge directly from cartographic experts, it is possible to analyse their results (maps). However, no clear method for analysing maps to provide this knowledge is presented and there still appears to be a heavy reliance on the subjective.

OTHER MAPPING EXPERT SYSTEMS

Expert systems and AI have been seen as being useful to many aspects of cartography and the mapping sciences. Two areas of particular note are the use of AI techniques in Geographic Information Systems (GIS) and in Remote Sensing.

More specific cartographic applications include various aspects of automated interpolation and contouring (Mark, 1986, Palmer, 1987), matching lines at sheet edges (Heivly, 1986) and accuracy assessment (Clark, 1987). Applications in GIS have been particularly with respect to data structures (Gahegan & Roberts, 1988, Peuquet, 1983, 1984, Ranzinger, 1985, Smith et al., 1987) and intelligent handling of queries to the system (e.g. Goodwin, 1987, Maggio, 1987, Menon & Smith, 1989, V. Robinson, 1988, V. Robinson et al., 1987, 1988, Stoms, 1987).

Remote Sensing has seen a particularly wide interest in AI and ES techniques, particularly in the interpretation of satellite imagery. This body of literature, which has expanded exponentially since the mid 1980s, is now probably larger than all other aspects of AI in the mapping sciences, but is beyond the scope of the present study.

CHAPTER FOUR

An Expert System for Cartographic Design

> Never admit to being an expert. An expert is someone who knows ***most*** of what there is to know about a subject . . . [1]

PROBLEM IDENTIFICATION

The need for expert systems in cartography

The previous chapter indicates that there is currently a considerable interest in developing expert systems for various aspects of cartographic design and production. Few of the reported studies discuss who the intended user of the system is or how the system will be applied in the map making process. An obvious starting point is to re-examine the needs for developing expert systems discussed in Chapter One, with specific reference to cartographic design, although as a first step it may be useful to eliminate areas of map design where expert systems have little application

There are large areas of cartographic design where expert systems currently have little to offer. For example, designing a topographic series is an event that takes place relatively rarely and the design adopted is likely to be used for many years, perhaps with some modifications through experience, changing requirements, or new production methods. Each series has to take into account many factors, many of which will not be the same as the topographic series required for a different country or different scale. The time scale for producing the design is likely to be relatively long and there will be opportunities to consult acknowledged experts and experienced map users.

Similarly, the design of products such as hydrographic and aeronautical charts has evolved to a highly refined state and there are international agreements on many aspects of the design of such products.

In these cases expert systems will have many more benefits in production rather that design, particularly in many aspects of compilation where smaller scale maps have to be derived from larger scale (digital) source data. Many of the systems discussed in Chapter Three address these needs, such as line simplification, name placement, etc. Other aspects include the selection of

[1] Poiker, T.K. pers. comm. 1985

soundings for hydrographic charts, contouring, selective omission of minor tributaries in river networks, etc.

The two situations where expert systems do have much to offer map design are (a) where the map is likely to be a 'one off' design and is wanted quickly and (b) where the map author is not an experienced cartographer and cannot 'imagine' what the final design will look like. In many cases both of these circumstances will apply.

Cartographic expert systems for whom?

Another important element is the level of cartographic knowledge of the map maker. Most serious expert systems have been developed to assist users with at least a basic knowledge of the system domain. Intelligent computer aided design (ICAD) systems are intended to assist designers in their normal job. Similarly several of the systems identified in Chapter 3 as aiding the map production process would be most beneficial to experienced map editors and cartographic draughtsmen. As already noted, however, an increasing number of maps are being produced by those who have no cartographic background, have some knowledge of the information they require to be illustrated and have access to a computer mapping programme or GIS which can produce maps of their data. In the past one might have expected these specialists to employ a cartographer to produce their maps, but now they can produce maps themselves with the same technical quality as a cartographic draughtsman by using computer plotters, etc., i.e. they have also taken on the role of the cartographer.

A group at even greater risk of producing inadequate maps are those with access to mapping systems who have limited knowledge both about the information and cartography. Current developments in information technology are allowing much wider access to large amounts of information. Encyclopaedias, atlases and large databases are now being promoted on CD ROM (compact disk read only memory). In theory anyone with a CD player attached to a personal computer can access the U.K. or U.S. census files and produce statistical maps of a wide range of attributes. Unfortunately, although many such products include mapping programs (e.g. Supermap), they offer no guidance in producing sensible maps with the data. Indeed, some packages *encourage the wrong choice* of mapping method and may not include generally accepted and appropriate methods for the data they contain.

The importance of the totally naive user should not be over-emphasised however. Despite the apparent popularity of the general idea, it is difficult to take seriously the concept that someone will suddenly decide to create a map of deaths due to bilharzia in West Africa, with no prior knowledge or understanding of tropical diseases, or mapping.

Thus, the group of users most likely to benefit from the development of a cartographic design expert system are those with some specialist knowledge about the information to be mapped, but with limited cartographic knowledge, or limited time to explore the options available for mapping the information. The extension of such a system to cope with less knowledgeable users relies more on developing explanations of the information than on developing the cartographic design aspects.

Choice of Subject

As stated in the Introduction, current advice on developing practical expert systems suggest that relatively well defined narrow domains should be chosen. They should be applied to subjects where there are human experts who regularly perform the task better than most other people. "Designing an expert system to add single digit numbers is silly, because almost everyone does this well. On the other hand, designing an expert system to predict the stock market is doomed to failure as no human expert does this consistently well" (Bahill & Ferrell, 1986; 50). Most expert systems have been developed for well structured problems which can be easily formalised. Design problems have proved especially difficult in this context. Design has been characterised as an ill-structured problem, " ... i.e. one which is difficult to formalize and difficult to solve, especially with man made problem solvers" (Begg, 1984; 45).

In identifying a narrow domain within cartographic design there are two limiting factors to be considered, map topic and scale. If the topics which can be produced are limited to a single subject or small group of related subjects, e.g. geological maps, or population maps, then specific rules for these maps at a wide range of scales could be developed. If a broader range of topics is desired, then to develop a practicable system the range of map scales considered must be limited. This limitation on scales is imposed due to problems of generalisation, many aspects of which have still to be automated satisfactorily.

Several options are worthy of consideration. There are many computer mapping packages designed to produce statistical maps at relatively small scales. Most of these have the capability of producing well designed maps, but there is nothing built into the system to assist a user with little knowledge or ability in map design. Most systems do provide default values, but often these are inadequate or even contrary to basic cartographic principles. Thus a cartographic design module added to such a system would be very beneficial.

The use of Geographic Information Systems for managing and mapping the natural environment is becoming more common and the maps produced in this field, which vary considerably in type and scale, would present an interesting challenge for expert system development.

As mentioned above, there is considerable interest in the development of databases based on CD ROM technology. This includes digital atlases where pre-designed maps are displayed on a computer monitor, but more interestingly, extensive databases of map-based information, such as the proposed World Digital Database for Environmental Science (WDDES) or the Digital Chart of the World (DCW). These include base information for mapping at a nominal scale of 1:1M. Users may add their own special topic information to this database. Cartographic publishers, such as Bartholomew and Lovell-John's, are also currently developing similar products, and this is seen as a growth area for cartography.

The map topics selected for this project are those that would be based on this last group of products. This includes maps of individual countries, parts of countries or groups of small countries found in regional atlases, such as those used for secondary education. An example would be the home country and regional maps in products like "A Senior Secondary Atlas for Nigeria" by Collins-Longman (1983). This type of atlas, while being in part a general world reference atlas, includes a number of maps devoted to a variety of special topics about the home country and surrounding region. It forms a suitable model as the types of maps and phenomena depicted are varied, testing the breadth of the system, but the maps are generally relatively clear and uncomplicated. Typically, multiple maps are used to show the wide range of phenomena, rather than highly complex maps showing many phenomena. Similar maps are also found in textbooks and atlases about specific countries and regions, so such a system would have wide applicability in the shift from printed reference material to the digital domain.

Scale for these maps ranges from about 1:2 000 000 to 1:15 000 000. The format of this type of product is also appropriate given the hardware limitations of most micro computer systems. These atlases are typically A4 size (about 30 x 20 cm) or smaller, and therefore maps larger than A3 size do not have to be considered. A common size for computer monitor displays is around 30 cm wide by 20 cm high. Thus, maps intended for hard copy output up to this size can be shown true to scale on the display; an A4 portrait (upright) image with no margins would be shown at about 75% of its true size and an A3 landscape image at about 50% of its true size. Other factors have to be taken into account, but it is at least practical to consider displaying these maps on standard computer monitors, and also to produce hard copy on inexpensive printers and plotters. Designing maps specifically for slides and overhead projection foils would be obvious extensions. For the prototype system however, only maps displayed on the computer monitor will be considered in detail.

In addition to small scale topographic maps, the type of atlas mentioned has a wide range of special topic maps, such as relief, land use, communications, climate, etc., which include point, line and area features and attributes in both numerical and non-numerical form. Thus rules will be required to deal with most types of information found on maps. It is intended that a database containing base data (i.e. topographic information) and special topic data will be part of the system. This will allow a variety of maps to be produced. Apart from this database, the map author must also be able to add special topic information for the system to be of more general use. To facilitate this, a complete system must be capable of interacting with the author to allow him to describe the nature of the phenomena to be mapped and the information available.

KNOWLEDGE ELICITATION

Having determined what a cartographic design expert system is for and who the intended user is, the next step is to gather the knowledge required to solve the problems involved and to enter it in the knowledge base in the most appropriate way to solve the problem.

Cartographic design, like many other areas of design has few formally stated rules to follow. Most of the research on cartographic expert systems seems to rely heavily upon written evidence in text books and 'scientific' journals. This is the easiest way of acquiring the basic knowledge required, but, as was noted in Chapter 2, it is rarely complete and texts intended for students do not always mirror the process as carried out by the expert.

The Cartographic Expert

If more direct input from experts is required, how are they identified, and how is their knowledge acquired? Identifying true cartographic experts is not an easy task. There are obvious examples, such as Imhof, but although one may identify well designed maps, frequently their designer is not directly credited. As a minimum, to be called an expert, one should have practical experience of applying the knowledge, but being an expert implies greater depth and breadth of knowledge than possessing the technical ability to carry out a skilled task.

The production of one map may involve a number of people. The map editor who compiles the information may be or become an expert in the information, must have some ability to judge how much information can be included in a given map and some appreciation for how it may be represented, but may not necessarily have the graphic abilities to design a 'successful' map. The cartographic draughtsman may be highly skilled, but his main task is in reproducing the information included in the compilation by following a set of specifications, and again need not possess the ability to design the map. This is not to say that neither of these groups of people should be consulted, nor imply that they may never be considered as being 'expert'. Indeed, in many cases both the compilation and production are carried out by one individual: if this person also carries out the design of the map and does this successfully on a regular basis then they must seriously be classed as being a map design expert. In the European context of the 'map editor' is often responsible for negotiations with clients on content and design; carry out any experimental work; compile the map; write the design specifications; draw the production flow diagram; and oversee the work to proof correction. Such individuals to be successful must be, or soon become, real 'experts'.

An obvious group to consider as being cartographic experts are academics and educators, particularly those with extensive publication records. Unfortunately, however, the ability to carry out research and write about map design does not necessarily imply an ability to practice map design, and many of the deficiencies noted in Chapter 3 are due to lack of experience in map design. There does appear to be a great emphasis on the use of academic publications as the basis for map design expert systems. In conversation with a researcher from a major cartographic expert systems project in the mid 80's, it emerged that the idea of discussion of the problems with acknowledged cartographic experts had not been considered, with published works forming the sole basis of knowledge elicitation.

The map as a source of knowledge

Even if cartographic experts are not directly consulted, they can be consulted indirectly through the artefact of their expertise, the map. Forrest and Pearson (1990) and Keates (1982) have emphasised the importance of studying existing maps as an aid to understanding map design (as opposed to understanding the phenomena depicted, i.e. map reading), yet this appears to be a minority view in standard cartographic textbooks. Muller (1990) based his generalisation system on an examination of a series of maps at different scales and Grossler (1993) carried out a comprehensive review of population change maps to help build rules for an expert system, so there has been some use of this approach.

One immediate difficulty would appear to be in deciding what is a good map, or how maps should be judged. It is unlikely that this will ever become anything other than a subjective process, although some guidelines could be followed. Obviously some appreciation of cartography, the processes involved and the possible alternatives that could have been employed is essential for meaningful evaluation. Some knowledge of the phenomena and data being depicted will also be beneficial. It must be remembered that the map is more than a decorative object: its function is to portray spatial relationships in a symbolic form.

Despite the apparent difficulty in assessing map design, there is obviously wide agreement amongst cartographers on maps that are well designed. A classic example is the Swiss topographic maps which are universally acclaimed for their depiction of relief. Evidence of this type of agreement was apparent in the British Cartographic Society 1990 design awards where a map designed by Geoprojects Ltd. won several awards, assessed by independent judging panels, and other maps by the same firm were highly commended in several categories. A similar pattern of events is frequently repeated.

THE EXPERTISE

The basis of knowledge elicitation for this project is introspection. To be appointed an expert by self acclaim may be viewed with derision, but it is expedient, and as long as operational use of the knowledge can be evaluated it serves a useful purpose. In this instance, it is based upon some 20 years of studying cartography, and longer in looking at maps. During this period some experience has been gained in designing maps both for print and for computer displays, although this is less than would be desirable. This knowledge is backed by reference to the cartographic literature. It is not possible to list every source consulted in developing this knowledge, although a review of the main sources is provided by the author in Forrest and Pearson (1990, see Appendix G). The primary source of textual knowledge is Keates (1989a).

Further support for the rules incorporated into the system is from the study of existing maps and atlases that cover the relevant scales and representations. Again the sources are too numerous to detail, but the main atlases consulted include Collins-Longman "Senior Secondary Atlas" for Nigeria (1983), Collins "Atlas of the World" (1984), Philips' "New World Atlas" (1988), Times Books/Bartholomew "The Times World Atlas (Comprehensive Edition)" (1986) and "The Ordnance Survey Atlas of Great Britain" (1982).

BUILDING THE KNOWLEDGE BASE

Cartographic design, like most design problems, is characterised as being an unstructured problem. In order to create a cartographic design expert system the first step should be to formalise the map design process, but like many other areas of design this has yet to be done. If this can be achieved then what initially appears to be an uncomputable problem can be at least partially solved if it is properly divided up.

This lack of formalism has not prevented cartographers from designing maps. Evidence in the form of published maps indicates that the practice of cartography is well known, even if the cartographers concerned have not started from a theoretical analysis of what they are doing[2].

There has been some interest shown in formalising the map design process. Eastman (1987) attempted to develop a "graphic syntax" for map design directed at expert systems applications, but fails to relate this directly to the actual process of designing maps. Mackaness and Scott (1987) attempted to define map design for expert systems. There is a brief passage on the 'conventional' design process, but this is not developed into a model for design by expert system. Aspects considered extend well beyond what might be considered the design process and discuss geographical knowledge and spatial cognition. They concluded that there is a wide range of aspects related to map design that need to be researched before any reasonable attempt can be made to automate the process, although they dismiss the notion of using expert systems to produce derived maps as 'simple', and concentrate on the processes involved in making the 'original' map.

This pessimistic view expressed by Mackaness and Scott (ibid) and by many cartographers when expert systems are proposed seems to stem from a lack of understanding of map design and expert systems. The apparent lack of written rules for cartographic design only causes concern if one considers the extreme range of possibilities for map topics and map scales. Once the scale range, location, subject and purpose have been established, the options are greatly reduced and there are many example maps which illustrate what can be achieved. That is, by moving from some vague notion of map design to the design of a specific map the problem of design becomes potentially solvable.

Thus, before attempting to develop a working system an attempt has been made to formalise the map design process. This is based upon an understanding of the information to be mapped and the processes involved in producing a map. These two themes are developed in the following chapter. Chapter 6 then builds on this model by codifying relevant cartographic practice in the form of verbal

[2] Keates, J.S. 1990 - pers. comm.

descriptions of the rules, conventions and processes required for the prototype system. These are then used to form the system's knowledge base.

CHAPTER FIVE

Geographic Information, Representation and Map Design

> The design process is a series of analytical and creative thinking steps that provide logical approaches for design solutions, helps the solutions meet [user] requirements, aids in the determination of suitability studies and serves as a visual and oral presentation basis.[1]

This chapter comprises two main sections. The first is a consideration of information for mapping and its representation; the second outlines the basic map design process and how it may be automated, in effect an outline specification of the system to be developed. In some cases, broad generalities are stated. These are not intended to be specific solutions to all situations that may occur in cartographic design, but descriptions of some of the problems that may face the proposed system and their possible solution. This is seen as a preamble to developing specific rules which are detailed in subsequent chapters.

PHENOMENA, DATA AND REPRESENTATION

Many of the writings on cartographic design expert systems immediately launch into the problems of symbolising the information. However, in a proper analysis of the map design process one must first of all establish some basic facts about the information to be mapped. Arguably, one should go even further back in the process and examine the reasons why the map is to be made in the first place, its intended user or uses and many other factors. Apart from a few very basic questions this is beyond the scope of the present study, but a full consideration of the nature of information which is to be mapped is appropriate.

There are four aspects to be considered here: the nature of the phenomena being mapped; the locational data available about the phenomena; the measurement of the characteristics of the phenomena; and their possible cartographic representations. Cartographic texts understandably concentrate on the last of these. Unfortunately, there seems to be no comprehensive study of the relationships between these aspects, although Peuquet (1984a & 1990) discusses some basic relationships in relation to establishing a framework for cartographic

[1] Hsu (1992) - on the role of design in landscape architecture.

data structures, and Keates (1989a) considers the direct influence of phenomena on map design[2].

Characteristics of Phenomena

Most authors (e.g. Robinson & Sale, 1969) consider three basic categories of phenomenon: points, lines and areas. To this, several add surfaces or volumes (e.g. Morrison 1974, Unwin, 1981)[3]. Strictly speaking temporal variation and movement should also be considered, but generally these will be omitted from this study, as is typically the case in cartographic literature. Although this is an apparently simple classification of the myriad possible phenomena, it is worth examining the four main classes in more detail. Fundamentally, however, the most important distinction is between continuous phenomena and discontinuous phenomena (Keates, 1989a) and this should ideally be reflected in their representation.

A phenomenon distributed at points appears intuitively simple with each occurrence being denoted by a set of co-ordinates. However, very few features actually occur at a point in the strictest sense of the term. Normally what we are considering are discrete features of some finite size, which are discontinuous in their coverage and which we deem to be points given the scale of the map. For example a factory can cover some considerable number of square meters on the ground. On a large scale plan it is be represented as an area enclosed by its walls, but on small scale maps it may well be shown by a point symbol representing the existence of the factory or by a symbol representing the value of its output. Another classic example of this is the treatment of towns and cities on small scale maps. Clearly these spread over some considerable extent, but for small scale mapping purposes we can consider them to be distributed at points.

Many kinds of discrete phenomena are seldom considered as occurring at points, such as population. "Generally they are not fixed in location, and can only be recorded as being present at or within a location at a given time." (Keates, 1989a; 205) Rarely do we have data on individuals. Normally they are enumerated

[2] In recent GIS literature the terms entity, attribute and object have become widely used. The terms used here are more in line with 'traditional' cartographic literature. Effectively, phenomenon and entity are equivalent, an attribute describes a characteristic property of an entity, which necessarily defines its level of measurement, and object refers to a digital representation of the entity as opposed to a graphical representation.

[3] It should be noted that there is some confusion in the literature about the use of the terms 'surface' and 'volume', with some authors such as Morrison (1974) considering a volume to be the difference (quantitative value) between a surface and a datum plane (actual or conceptual) and others using the term surface to refer to essential the same thing e.g. Unwin (1981).

by some area and may be represented by area or point symbols depending upon map scale, purpose and design.

Linear phenomena by their very nature must be continuous, (in space if not in time) although often what we consider to be linear features are actually zones of transition between two surfaces, such as the coastline, and indeed most 'natural' boundaries are of this type. They may also be tangible features on the ground such as fences or walls, but again our treatment of many features will depend upon map scale, e.g. a road at large scale may be depicted by two bounding kerbs representing its physical extent, whereas at small scale it is its importance as a line of communication that is mapped. Thus we may have both a change of concept from area (space between kerbs) to line (nominal centreline) and of representation (area to line). This is part of the generalisation process.

Phenomena occurring within specific areas may be present continuously over the whole surface, such as geology, soils, etc., or may be discontinuous, such as lakes, built-up area, etc. Effectively the latter dichotomous type is a subdivision of the first type where we have a binary division of the surface, rather than a more numerous set of classes and sub-classes. There are also cases where large parts of the map area may be 'unclassified', e.g. residential building type. Again this can be considered as a further refinement of this general class.

Surfaces refer to those phenomena which are continuous and vary quantitatively in space, expressed as a variation in value above some datum, the topographic surface being the obvious example. Some phenomena vary continuously both spatially and temporally, but variables such as temperature and pressure, if considered at some specific level, e.g. ground level, may also be treated as surfaces. Precipitation, while not continuous, may also conveniently be grouped here (Keates, 1989a; 205) as we are normally dealing with averages over time.

Volumes are three dimensional entities bounded by a set of surfaces. If we adopt a fixed datum and measure the distance or value of one of the bounding surfaces from this we can simplify this to be considered a surface phenomenon (see above). Volumes will not be considered further.

For clarity, the terms discrete, linear, specific area(s) and continuous surface will be used subsequently to describe the distribution of phenomena to distinguish them from data or symbol types.

Relationship Between Phenomena and Locational Data

Before one can commence to design a map one must have data to map. This data will in some way, although not necessarily simply, relate to a phenomenon or phenomena which has in some way been measured or sampled. While a phenomenon may be distributed at discrete locations, along lines, contained in specific areas, or vary continuously over a surface, spatial data can only exist in the form of points, lines, or areas[4]. Areas must either be defined by their outlines, which implies there must be line data, or as some regular tessellation of the surface (e.g. grid cell data)[5]. Area boundaries may be the actual outline(s) of the phenomenon, or some imposed (often arbitrary) boundary.

To use the computer data structure analogy, these classes of data are 0-dimensional, 1-dimensional or 2-dimensional data. Despite this seemingly simple classification of phenomena and data, frequently the data available about a phenomenon has not been collected for the purpose of mapping and may not reflect the actual distribution of it. This situation may also be compounded by the available data being of a secondary nature, having been pre-processed for a variety of reasons.

While there is an unlimited number of phenomena that may be mapped we can identify a small number of frequently recurring combinations of classes of phenomena and the spatial data available about them. These are illustrated in Table 5.1, from which 8 primary combinations emerge:

1. Discrete units, specific (point) location data
2. Discrete units, data aggregated by bounded area
3. Discrete units, data aggregated by cells
4. Linear phenomenon, with line data
5. Specific areas, outline data
6. Specific areas, cell data
7. Continuous surface, data sampled at points
8. Continuous surface, data sampled along lines.

In addition to these primary combinations, there are several possible secondary or derived distributions. For example, frequently information about specific areas is attributed to a single point within the zone, usually its centroid. Also, any set of numerical point data may have isolines produced from it. Thus we may have discrete units aggregated by specific areas, data attributed to a point within each unit, and isarithms interpolated from these. Other such complex permutations are possible, but there may be occasions when the map author may

[4] Three dimensional co-ordinates (X,Y,Z) may be given, but generally the third (Z) co-ordinate can be considered to be an attribute.

[5] Three dimensional tesselations of space are not considered.

only know the nature of the phenomenon and the form of data available, but not how the data has been derived.

Table 5.1
The relationship between phenomena and locational data (frequently occurring combinations).

Data	0	1	2	2
Dimension			bounded	tessellation
Phenomena				
Discrete units	P	S	P or S	P
Linear	-	P	-	-
Specific areas	S	-	P	P
Continuous surface	P	P or S	-	-
	P = primary data			
	S = secondary data			

Level of Measurement

In addition to the locational characteristics of the information we can also classify its attributes. When information is gathered, measurement is the process of assigning a class or value to the observation. This attribute of a feature may be either numerical or not. It can also be seen that non numerical attributes can either be simply describing differences in kind or types of features, or can indicate ranks or hierarchies of features. This characteristic of information is known as its level of measurement (Unwin, 1981, Robinson et al., 1984). The conventional classes are nominal, where there is a change in type or kind, ordinal where there is a ranking of the data, and interval & ratio where some numerical value has been measured or calculated. Morrison (1994) also usefully identifies dichotomous as a sub-class of nominal and further expands the range of possible data types to include the relationships between sets of points and sets of areas. Combinations of these classes are possible for some phenomena. For example, power stations may be distinguished nominally by their method of generation and on a ratio scale of their output capacity. Knowledge of the level of measurement together with the nature of the phenomenon and the spatial data type will allow one or more cartographic representations to be assigned to the information. Figure 5.1 illustrates the relationship between locational data and level of measurement and shows some typical representations.

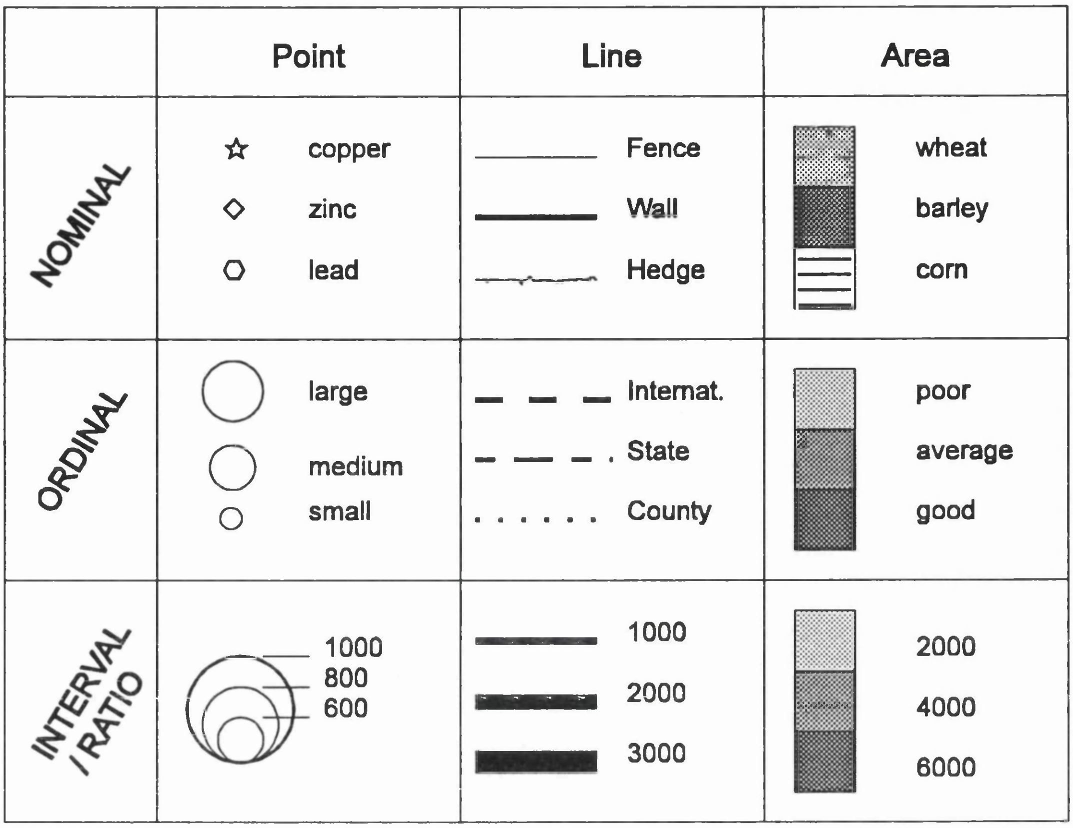
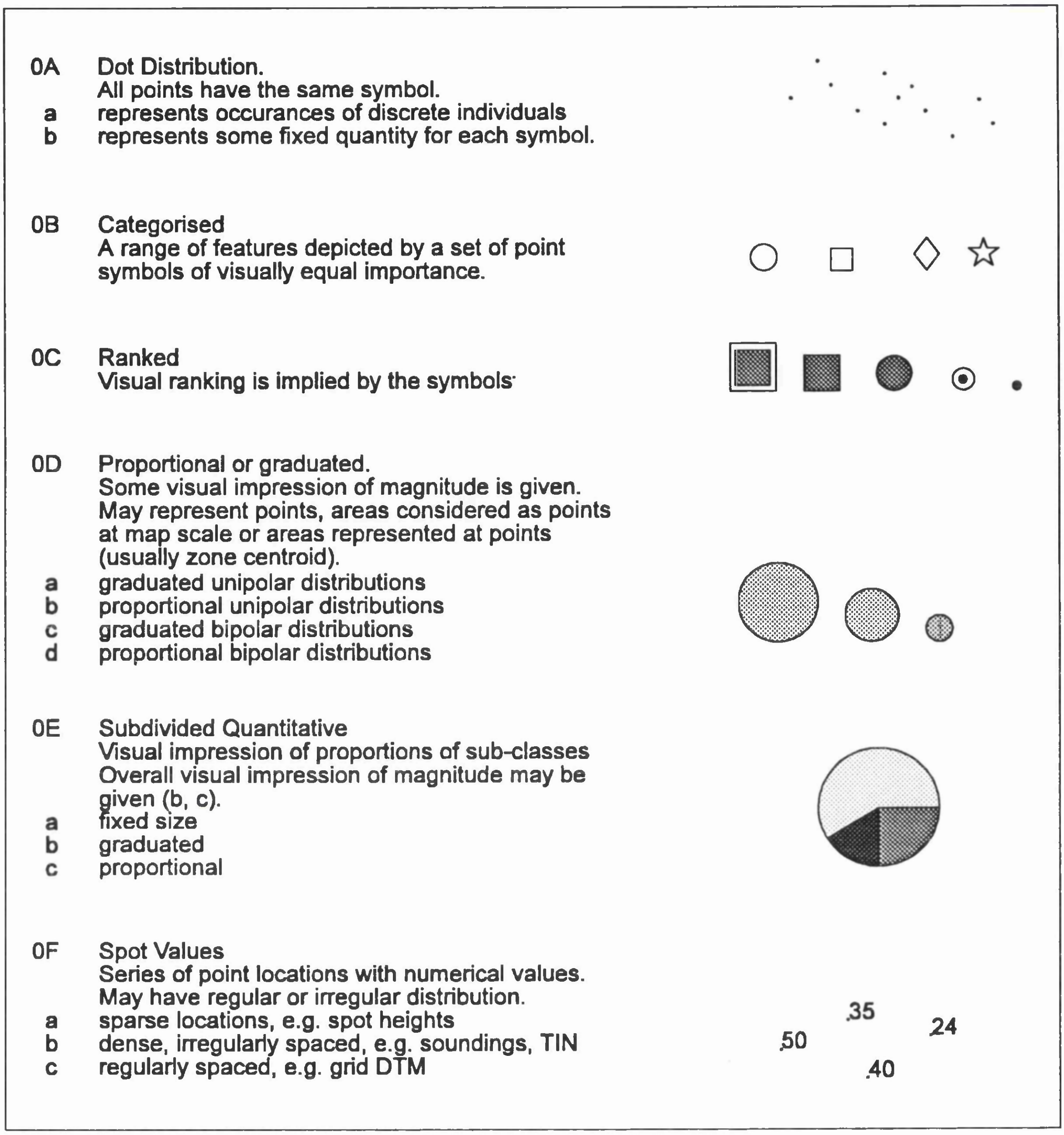


Figure 5.1
Examples of symbols for different measurement levels

Cartographic Representation

Despite the wide range of phenomena that may be mapped, there is a limited number of cartographic representations available. Figure 5.2 a, b, c & d outlines these. Frequently it will be possible to depict a data set by more than one of these methods, although once the purpose of the map has been determined and the other information to be included has been decided, the choice is often limited. Where there remains a choice, the map author may decide which representation to use or accept the default option.



0A Dot Distribution.
All points have the same symbol.
a represents occurances of discrete individuals
b represents some fixed quantity for each symbol.

0B Categorised
A range of features depicted by a set of point symbols of visually equal importance.

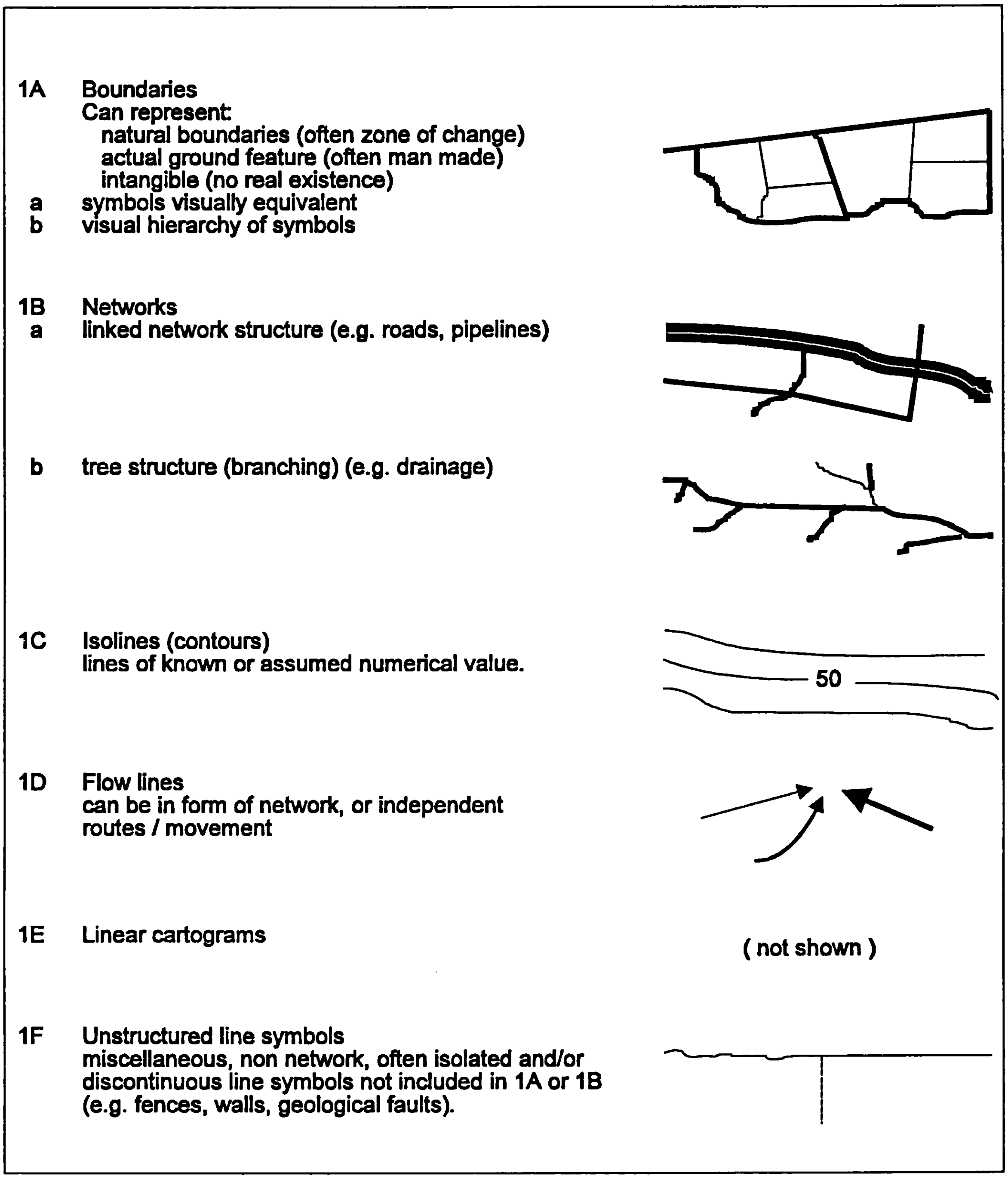
0C Ranked
Visual ranking is implied by the symbols·

0D Proportional or graduated.
Some visual impression of magnitude is given.
May represent points, areas considered as points at map scale or areas represented at points (usually zone centroid).
a graduated unipolar distributions
b proportional unipolar distributions
c graduated bipolar distributions
d proportional bipolar distributions

0E Subdivided Quantitative
Visual impression of proportions of sub-classes
Overall visual impression of magnitude may be given (b, c).
a fixed size
b graduated
c proportional

0F Spot Values
Series of point locations with numerical values.
May have regular or irregular distribution.
a sparse locations, e.g. spot heights
b dense, irregularly spaced, e.g. soundings, TIN
c regularly spaced, e.g. grid DTM

Figure 5.2a
Cartographic representations - points



1A Boundaries
Can represent:
natural boundaries (often zone of change)
actual ground feature (often man made)
intangible (no real existence)
a symbols visually equivalent
b visual hierarchy of symbols

1B Networks
a linked network structure (e.g. roads, pipelines)

b tree structure (branching) (e.g. drainage)

1C Isolines (contours)
lines of known or assumed numerical value.

1D Flow lines
can be in form of network, or independent routes / movement

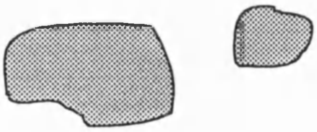
1E Linear cartograms

(not shown)

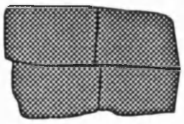
1F Unstructured line symbols
miscellaneous, non network, often isolated and/or discontinuous line symbols not included in 1A or 1B (e.g. fences, walls, geological faults).

Figure 5.2b
Cartographic representations - lines

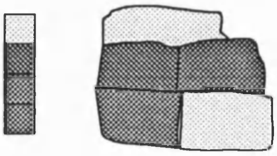
2A isolated areas
binary (dichotomous) division of the surface
- special case of 2C



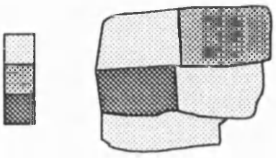
2B Unclassed
-simplifed case of 2C, e.g. 'political' map
a single visual level - al symbols equivalent
b hierarchy of visual levels (e.g country/state/county)



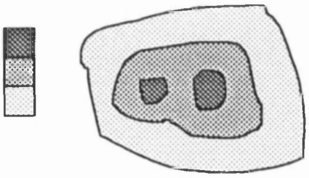
2C Chorochromatic (colour patch, mosaic or categorical)
several instances of same class, classes equivalent
a single visual level (uniform)
b hierarchy of visual levels (i.e. classes and
sub-classes)



2D Graded series
two possibilities for boundaries, but treatment of
symbolisation the same
- Choropleth - delimited zones imposed on
distribution (normally administrative)
- Dasymmetric - delimited zones with boundaries
derived from distribution
a Unipolar variation of symbols, implying quantities
b bipolar variation of symbols around neutral
c bivariate symbolism (not dasymetric



2E Layered colour series
graphically the same as 2D, but different spatial
distribution.
a unipolar variation in symbols, implying quantities
b bipolar variation (e.g. temperature zones)



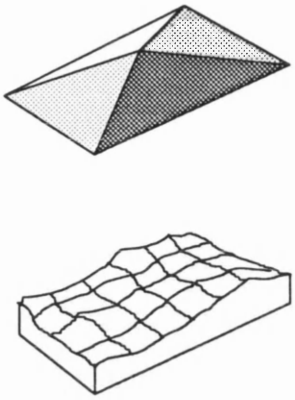
2F Hypsometric and Bathymetric colours
Spatially the same as 2E, but different (specific)
graphical treatment.

(not shown)

2G Area Cartogram
areas scaled by some quantity, not true extent

(not shown)

Figure 5.2c
Cartographic representations - areas



3A Shading
e.g. hill shading

3B Block Diagrams (2 1/2 D views)
(arguably not a true map - non orhtogonal)

3C 3D Surface Model
(cannot be displayed in this form - an internal representation in a computer or a physical model)

Figure 5.2d
Cartographic representations - Surfaces

Having classified phenomena, data and level of measurement, Table 5.2 illustrates the relationships between these and possible cartographic representations. It is planned to develop a separate expert system to classify information the user may wish to add to the database. This would take the form of a relatively simple classification type of expert system, which could run independently or as a sub-system of the main package.

Table 5.2

Relationship between phenomena, data & representation

Phenomenon distribution	Locational data dimension	Level of measurement	Possible representation methods
discrete	0	nominal	0B, 0A
discrete	0	ordinal	0C
discrete	0	interval/ratio	0D,[1C,2E]
discrete	1	ord./int./ratio	1C,2E
discrete	2	ord./int./ratio	2D,0D,[0A,1C,2E]
linear	1	nominal	1A,1B
linear	1	ord./int./ratio	1B,1D
specific areas	0	interval/ratio	0D,[1C,2E]
specific areas	2	nominal	1A,2A,2B,2C
specific areas	2	ord./int./ratio	2D,0D,[1C,2E]
cont. surface	0	interval/ratio	0F,[1C,2E]
cont. surface	1	interval/ratio	1C,2E

Notes: 2 dimensional data is in the form of boundaries (area outlines)
[ ] - requires further processing or information for this representation

THE DESIGN PROCESS

The general procedure for designing a map follows the route of: Compose; Compile; Symbol specification; Produce; Adjust. A similar route can be followed with computer aided cartography, and indeed it is logical that an expert system follow a similar route to a Cartographic expert. The stages to be followed are: Description; Layout; Data Selection; Symbolisation; Display; Modify. These are described in more detail below.

Description

In this step the aim is for the user to describe to the system some basic information about the map required.

First then, the user must inform the system of the type of map to be produced. Immediately, a distinction can be made between topographic or 'general purpose' maps and special topic maps. A topographic map ideally shows all information with the same level of importance, i.e. no one aspect of the map should dominate, although in practice cultural information tends to dominate on most topographic maps, and in any good design there should in any case be several 'visual levels'. For a special topic map, the special topic information normally will be the dominant part of the graphic image with the base information providing context and orientation for the map user. Thus, the system will have to know the topic of the map at an early stage.

The system will 'know' about a definitive number of map topics that can be produced from the information in its database, but the user may not be familiar with such titles and may require help in defining what he wants. He may indeed require a map that the system does 'know ' about, but refer to it by a different title. From the author's description of the information to be included in the map the system should be able to determine the type of map required. The user's name for this may be added to the knowledge base for future reference.

If a topographic map is required then the system will be able to exercise almost total control over the map design as all the information will be contained in the system data base and the system 'knows' about this type of map topic. If a special topic map is required, more information will be needed from the user, particularly if the main information to be mapped is not included in the system data base. In this event the user will have to describe the phenomena to be mapped, the data available, and supply the necessary spatial and/or non spatial data.

The purpose for which the map is to be used is also important in determining what must or may be included in the map, the level of detail which may be required and hence the scale that will be required to show the desired detail. As the system will be specifically limited to producing small scale maps in a particular scale range, there are obviously limits on the intended use of the output. Clearly maps at these scales are not intended for detailed measurement, but more for providing general information about an area or an overview of specific distributions in an area.

The intended user of the output should also be considered at this stage. The map author can also be the map user, but the map may be being produced for a wider range of users. If the author is also the intended user then one can normally assume some familiarity with the location or information being portrayed. If the map is intended for others they may or may not be knowledgeable about the area or subject. If the map is intended for naive users it may be desirable to make the map simpler than for an expert user. This will reflect on both the content of the map and the level of detail at which each element is shown.

The form of output required may impose limitations on the cartographic possibilities. For example a monochrome map generally cannot show as much detail as a colour one; screen resolution is usually much lower than that of lithographic printing; maps for showing as slides generally need to be quite simple.

The desired level of detail required on the map will influence the amount of information selected and also the level of generalisation used. It is closely related to the map purpose, the form of output and to the scale of the map, scale probably being the most important limiting factor in the actual amount of detail that can be shown.

Layout

The factors to be decided at this stage are the actual geographical area to be mapped; the format (i.e. height and width) of the output; and the scale of the output. A decision on the first of these and one other will determine the third. The level of detail determined above may influence the choice of scale. Some backtracking may be required if it is not possible to show the desired area in the available format at a scale commensurate with the topic or the level of detail requested.

Location. As a general point, the map author will know within reasonable limits the area to be mapped, although the size and shape of the area of primary interest and the purpose of the map may influence how much of the surrounding area should be included. It is unlikely that the system will be of much assistance in determining this, although an interim plot of the area may help the author.

Format. If the map is only to be displayed on a CRT display then the format will most likely be the maximum size allowed on the display, or that portion of the screen reserved for showing the map. If hard copy is required this will be limited by size of printer or plotter to be used, but may also be limited by other factors, such as the page size of a publication. If the map is for reproduction the author may have fixed criteria. If the author needs help at this stage then some basic rules of appearance can be used, such as ratio of sides etc., and selection of portrait or landscape orientation depending on the shape of the area being mapped.

Scale. The general principle with atlas maps is to use the maximum scale possible (to the nearest round figure) within the given format, so this will normally be calculated by the system based upon the size of the area and the format. It is possible that the user will require the map to be of a certain scale (if one of a series perhaps), thus once scale and location have been specified the required format can be calculated. Scale is a topic that is frequently misunderstood by map authors and users, particularly at the small scales used here so it is likely that the system will have to provide a considerable amount of explanation when it is being determined.

Marginal Information. The standard layout for the marginal information is to have the title, scale and legend placed outside the map neat line, with the title and scale across the top of the map and the legend on the right hand side. The user should be able to 'switch off' the legend on the screen to allow a larger map area to be shown, or a larger scale to be used.

Beyond this, the proposed system will not at this time make any attempt to design the layout of marginal information such as titles, legends, etc., to make the best use of the space available, although the inclusion of a few alternative layouts would be a trivial addition.

Reserving space for marginal information will influence the scale or format of the map, but will be the authors responsibility if they are required to be located elsewhere. Future developments could include assessment of the outline of the map area, the amount of legend space required, etc., and the suggestion of possibilities to the map author.

Data Selection

In producing a map conventionally, the selection of information and its classification is normally part of the compilation process. In a digital mapping system the data has in effect been compiled when it is entered into the database. What an expert system must do is extract the appropriate information from the database as it is likely to contain data on more phenomena than can reasonably be shown on a single map.

The information to be included in a given map will depend upon the type of map, the scale, and the level of detail required. Table 5.3 lists the information to be included in the system database. For this data to be used in a flexible manner knowledge about it must be incorporated into the system. This metadata could be included in the map design knowledge base, but more usefully it would be a separate knowledge base linked to the database which would allow easier integration of different databases. In the longer term the development of comprehensive metadata is seen as critical to the widespread application of expert systems to cartography and GIS.

General Maps. If a topographic or outline map is required then, as all the information will be contained in the system database and the system will be 'familiar' with the selected map topic, the system will be able to exercise almost total control over the selection and representation aspects of design. For a topographic map all the Information described as 'basic information' in Table 5.3 would normally be included, although the user could be given the choice of a 'physical' type map which includes hypsometric tints or a 'political' type map which has coloured administrative zones. The level of detail at which each feature is depicted will however depend upon the scale, the purpose and the specified level of detail of the map.

Special Topic Maps. If a special topic map is being produced, more information will be needed from the user, particularly if the main information to be mapped is not included in the system data base. Maps whose topic is one of those listed as supplementary information in Table 5.3, may require more user input than general maps, but will require considerably less than for special topic information supplied wholly by the map author. The basic cartographic representations of the information in Table 5.3 will be known to the system, as well as what base information is normally included in a map of the selected topic (i.e. there will be information about this in the knowledge base).

If information to be mapped is not in the database the user would be prompted to describe the phenomenon and the data he has available at this stage and build up a file of metadata. This can in fact be seen as a separate task to the design aspects of the system and fits the model of a classification expert system.

Table 5.3
Proposed database contents

TOPOGRAPHIC BASE INFORMATION

Political/Administrative Boundaries - International and internal (2 levels if available). These will be used with separate census data file for statistical maps.

Coastline - similar level in hierarchy as International boundaries. Two levels of generalisation should be available.

Drainage - network classified with at least 3 levels.

Lakes - large lakes (greater than 2mm^2 at the largest map scale). Must be linked with drainage network.

Railways - one level.

Roads - classified as highways/motorways, major roads, other roads.

Settlements - (administrative definition) database would contain classification based on simple hierarchy, e.g. National capital, State capitals, other cities and important towns, giving 4 levels of hierarchy.
Ideally settlements would be chosen from a separate database with a variety of factors, e.g. population, political status, remoteness, etc., with a ranking calculated from these. Cut-off point determined by scale initially. User could specify number of settlements to be included, or selection criteria based on facts in database and system make choice. Different default parameters could be specified for different map types. Number of categories dependent on scale, number of settlements to be included and map type.

Contours - frequently shown on atlas maps with non uniform interval, as basis of layer colours. Database should include all contours based upon minimum interval appropriate for region. Actual intervals used selected automatically depending upon scale. Area symbolisation may be included depending on the type of map.

SUPPLEMENTARY INFORMATION

(Information frequently used for special topic maps in regional atlases)
Geology, Soils, Land Use, Land Cover (Vegetation), Precipitation, Temperature. Census data, including population etc. (to be used with administrative boundaries and settlements above). Economic data (ports, industry, etc.)

While perhaps to be avoided for reasons of simplicity, it is possible that a map author will require a map showing more than one special topic. (The use of bivariate mapping of statistical/census information is not included in this discussion.) The map may for example show both temperature and precipitation. As there are strict limits on the number of continuous phenomena that can be shown by area symbolism - probably two at most, one being shown by area colours and the other by area patterns - the user will have to prioritise the phenomena to be depicted, and may have to opt for line or point symbols to depict some continuous phenomena. For example, a map could show annual precipitation by layer tints and January and July isotherms by two sets of lines.

Base Information for Special Topic Maps. In designing a special topic map one must have an appropriate base map on which to display the information. There are two common approaches to this. The first is to take a topographic map and reduce it to a background image, often by printing it in grey. This will mean that much superfluous information will be included and also that some essential information may be obscure.

The second approach is to design the base specifically for the map. This involves selecting the appropriate information from the topographic base and symbolising it to complement the special topic information. This should result in a better solution, and is the approach adopted here.

Frequently little consideration appears to be given to the level of detail of the base image when compared to the special topic representation. For example, a map with a very detailed coastline showing very generalised climatic information can mislead the user into thinking the special topic information is as detailed as the topographic information. Thus, some attempt must be made to have commensurate levels of detail for different elements of the map. This may involve simplifying the base information to reflect more closely the detail or accuracy of the special topic information.

Map complexity. Although the level of detail, map purpose and scale together will provide some indication of how much information should be included in the map, some problems will only emerge after the data has been (provisionally) selected. For example if a coverage of areas is to be included the system should check to see that the polygons are large enough to be perceived. If not, a more generalised representation must be used. Similar tests can be done on total length of line and number and average spacing of point symbols. Although this is not a true measure of complexity as it doesn't take distribution into account, it provides an initial indication to the system that the map may be too detailed.

Interaction of representations. As noted above, it is generally not possible to show many sets of area symbolisation. Some areas may have to be implied by their boundaries. Problems of spatial conflicts have been deal with in some detail by Mackaness (1986) and Mackaness & Fisher (1987) and solutions developed therein could be incorporated. Generally, and whenever possible, problems should be avoided by not selecting too many classes of information that normally require area symbolisation over the whole map area. Other conflicts will normally be resolved at the symbolisation stage by selecting symbols with sufficient contrast.

Generalisation. This creates many problems in map design, particularly as scale decreases. As discussed in Chapter 3, there have been several studies on expert systems applied to map generalisation and it is beyond the scope of this study to incorporate all the possibilities. Given that the range of scales available to the system is limited, generalisation can be resolved partly by selection and, where appropriate, by having two sets of linear data or coding linear data so that it can be produced at two levels of generalisation, determined by scale and level of detail required. Automated generalisation is not a feature of the proposed system.

Symbolisation

Having made an initial selection of the information to be included in the map (which may have to be modified once the map has been displayed) each element of the map will have to be given a symbol specification. The actual details of this will vary quite considerably depending on the phenomena being mapped, the scale, etc. The first step is to assign the cartographic representation to be used (Table 5.2). Each of the data sets to be included in the data base may be assigned one or more possible representation(s) based upon the nature of the phenomena, the locational data and its level of measurement as discussed above.

This is only the first step in specifying the symbols. Once the type of representation is known, specific symbols will have to be assigned to the information. In some cases this will be trivial, such as specifying colour and gauge of rivers. In other cases considerable effort will be required to select the most appropriate set of point symbols or area colour scheme, for example. Rules for the representation and for the data set will be used to narrow the choice, but inevitably user choice will play a major role here, at least in approving defaults, or choices suggested by the system.

Display

Having determined the information to be included and its graphical representation, the map can be displayed on the screen. This is largely a procedural task for the system, although, due to the nature of computer graphics and some of the representation methods, consideration will have to be given to the order in which items are drawn. Generally speaking area symbols will be produced first followed by lines, then points. More sophisticated measures will be required in many cases for hard copy output, in particular the masking of underlying symbolisation so that subsequent features are visible.

Modify

It would be ambitious to suppose that the first attempt at designing the map will be exactly what the system user requires, therefore the system should be able to interact with the user to modify any of the decisions previously made. This is similar to what a cartographer would do: preliminary designs may be reviewed for their effectiveness, or presented to the map author for approval. Any modifications requested would of course have to be processed through the knowledge base, and the user notified of any consequences. The system will store parameters for completed designs so that it is possible to backtrack should the modification not result in an improved map.

The biggest difficulty here however, is in assessing what good design is, as this is largely subjective and attempts to quantify this (e.g. Mackaness et al., 1986) bear little resemblance to the user's perceptual response. It is also arguable that the intended user of a cartographic design expert system is unlikely to be able to pinpoint what the design problems are, far less quantify them, therefore modifications are more likely to affect **what** is shown, rather than **how** it is shown.

This latter section on map design is further expanded in Chapter 7 to provide a full functional specification of an expert system for the design of small scale maps.

CHAPTER SIX

Developing Rules for Map Design: A Functional Specification of a Cartographic Design Expert System

> The lack of formal rules for map design is not simply a consequence of cartographic incompetence, or a lack of interest in the map user. It simply reflects the sheer difficulty of deducing a set of rules, capable of universal application. ... The complexity of obtaining information from a map is matched by the complexity of creating it.[1]

To cover every aspect involved in developing a cartographic design expert system, one would have to write what amounts to a textbook on cartography. The intention here is to discuss the main factors involved in each situation and note the rules that are relevant in developing the prototype system. The discussion concentrates on the requirements for small scale maps. In some instances the extensions required for scales beyond those proposed for the system are noted.

Although the steps involved are described here in a sequential manner, there will be cases where there is interaction between the main sections of the system described here and in Chapter 8, and where an iterative approach is necessary. For example, the amount of information selected is based upon the purpose, level of detail and scale. If too much is selected initially, then one or more of the earlier variables may have to be altered. Another situation which may arise is at the symbolisation stage when conflict arises between the representation of data sets, when it may be necessary to re-evaluate the selections made.

The emphasis of this chapter is on a verbal description of what will later be specifically coded into the system. For each aspect there is a brief description of the situation and what is required to solve it. In some cases this discussion is summarised by a list of 'Factors'. This shorthand form has been adopted in order to keep the discussions as brief as is practical. The discussion on each aspect is concluded by a list of the 'Rules' that need to be built into the system. The statements under the Rules heading include rules stating basic cartographic design principles, facts and definitions required by the system, queries for more

[1] Keates, 1982; p.113.

information and operating instructions for both the system and the system user. Unless the user seeks further information by using the help and explanation systems, all he will be aware of is operating instructions and questions which the system cannot answer from its knowledge base.

For convenience, the 'Rules' in each section to be incorporated into the knowledge base are numbered. An * indicates that the rule is specific to this system, rather than a general rule or statement, there being some form of limitation imposed by the system on the possibilities. Unnumbered rules (shown by '-') will not be included in the knowledge base, but are included here for completeness and to indicate possible future developments. Some of the 'rules' are statements of facts that apply to the system or the task and always apply; others are conditional and only apply when the condition is met. At this stage, confidence factors have not been included as part of the knowledge, although they are included in the system where appropriate.

In order to simplify the structure of the 'rules' set out in this chapter, several key words, etc., are used. These are explained below:

IF .. THEN .. (ELSE)	general format of a rule
AND, OR	logical conjunctions
ASK	get information from user
MENU	a list of choices is available. Options, if given, enclosed in {}.
CONFIRM	Items or values either set or selected are shown (verbally) on the screen. The user is asked for confirmation.
[ ]	default value enclosed

Generally, before moving from one module to the next the information gathered or set in that module will be displayed and the user asked to confirm that this is correct or what is desired.

It is important to note that the functional specification (this chapter) was written before the programming reached an advanced stage, i.e. the functional specification was developed before the program and not afterwards. Some subsequent editing has taken place to remove errors and inconsistencies or to clarify points, but these changes have been relatively minor. This does mean that there are some differences between what is suggested here and the actual implementation discussed in subsequent chapters, but the functional specification is retained in its original form so as to allow different implementations of various aspects to be attempted. One obvious difference is that in several places only the automated solution is incorporated into the system as it is this that is under test in the prototype system.

DESCRIPTION

Several factors have to be decided at this stage: the topic of the map to be produced; the purpose of the map; the intended map users; the level of detail required; and medium for final output. It is likely that the range and type of questions asked at this stage will be a major part of future developments of the system. A natural language interface would be particularly relevant for this module.

What is the topic or theme of the map?

Initially only the primary theme may be defined, but this may be extended later to allow secondary and tertiary themes. The map author can either enter a name which the system will match against list of known map topics (themes), or a menu of known map topics can be displayed.

If the map topic is unknown to the system, the user will have to describe his requirements to a greater extent. He may however be able to state that it is similar to a 'known' map topic. At the very least the basic 'class' of map must be stated. The three primary classes are : (1) basic, which includes outline and topographic maps; (2) cultural, which includes population and economic maps (mainly statistical); and (3) physical, such as climatic, relief, soils, etc. The class helps in identifying the type of base information most likely to be appropriate for the map.

In addition, the user may have to supply special topic data to be incorporated with basic data from system. A facility must exist for describing user data in terms of basic phenomena and information types to allow appropriate rules for its incorporation and representation to be applied. As mentioned earlier, this function will be performed by a related expert system.

RULES

1 System user must specify map topic (MENU available)

What purpose is the map for?

RULES

1 Choose between overview and analysis [default = overview]

Who is the intended map user?

RULES

1 Choose one of general users, knowledgeable users, map author (map author = knowledgeable user for the purposes here)

Output media?

This will control various factors, such as size, resolution, colour availability, etc. Initially only coloured screen output will be considered.

RULES

1 ASK what output media [Screen], MENU {Screen, Hardcopy, Slide, Overhead}

2 ASK if monochrome or colour required.

What level of detail is required?

A ten point scale is used with 1 = low detail and 10 the most detailed. Based on user, purpose and output media the systems suggests the appropriate level of detail to the system user. This may be accepted or modified. If modified, the system will ask for confirmation of any change varying by greater than 2 from default. Note that for screen based maps 8 is the maximum level of detail suggested. Initially a simple set of facts in the knowledge base will suffice, but in the longer term an algorithm could be developed which would allow more flexibility.

RULES

```
1   IF Purpose = Overview AND User = General AND Media = Screen THEN 2
    IF Purpose = Overview AND User <> General AND Media = Screen THEN 4
    IF Purpose = Analysis AND User = General AND Media = Screen THEN 6
    IF Purpose = Analysis AND User <> General AND Media = Screen THEN 8
2   CONFIRM level with system user. IF change > +2 OR change > -2 then
    ASK for confirmation
```

LAYOUT

The three decisions required in this module interact with one another. The user will have to specify the desired area to be mapped. Once either the scale or format are specified, the other can be calculated.

Location

There are a number of ways by which location could be specified. These include area name, by latitude and longitude limits, by projection co-ordinates, by specifying a number of places that must be included, or by graphical methods.

The use of names other than demarcated areas such as countries poses problems of interpretation. What is the extent exactly of 'West Africa'? These general terms may however be useful as a first stage in delimiting the area, forming the basis of an interim plot allowing co-ordinates to be given or a graphic definition to be indicated.

Graphical methods are where an outline plot of some large area (e.g. the World or a continent) is displayed and the user uses the cursor or a mouse to indicate a box on this. This may need several iterations as successive interim plots zoom in on the area of interest.

The use of different projections creates some difficulties as large regions will vary in shape on different projections. Taking this into account is beyond the scope of this study. As the test area is near the equator a simple Lambert's Cylindrical Equal Area projection will be used which allows simple conversion from latitude and longitude to xy co-ordinates for plotting. An equal area projection is used as equivalence is desirable for many of the topics to be mapped.

RULES

1 Location must be specified

2 IF location is adjusted THEN check scale and format

Format

The default format is the maximum size possible on the screen. Standard dimensions such as those for A4, A5, etc. should be built in to the system, both for full page images and layouts with a margin outside the neat line.

The interaction with location and scale must be checked any time there is a change in format.

RULES

1 IF know location and scale THEN calculate format ELSE ASK user for format [default = fill screen]. Menu available, one option is to calculate from scale.

2 Check that format fits output device

3 IF format changed THEN update scale and report this.

Scale

To simplify matters automatically selected scales will be limited to a predefined set of scales varying between 1:2 000 000 and 1:15 000 000, although the user may choose any scale within this range. The default scale is the maximum possible for the specified format, rounded to the nearest available smaller scale in the list.

The interaction of scale with location and format must be checked any time there is a change in scale.

RULES

1 IF know location and format THEN calculate scale ELSE ASK user for scale [default = maximum possible]. Menu available, one option is to calculate from format.

2 IF scale specified by user THEN check that scale fits output device

3 IF scale changed THEN check format and report any conflict.

SELECTION

From known values for level of detail and scale a 'selection index' is calculated (see Chapter 8 for formula). This is used as the initial basis for selecting what information is to be included in a given map. This is not a true indication of how complex the final map may appear, but determines which classes of information are to be included. The calculation used is derived empirically and alternatives could be substituted. This is a simplistic solution and is viewed as a first attempt at evaluating how much information to include. The ability to test more sophisticated methods will be incorporated into the system and should prove valuable in future developments and allow further research in this important area.

The selection index will be in the range 1 to 10 with higher values indicating less information is to be included.

Each known map topic will have a list of scores for each base information class and sub-class. Each of these scores has a value from 0 (do not include) to 10 (must include) indicating the priority of including the respective dataset in that map type. From this, a list of recommended datasets will be constructed by selecting those datasets which have a score equal to or higher than the selection index value. (Those with a score > 0 for the map type could be entered in an 'optional' list allowing the user to select these if desired, or to assist later with modifying the design specification).

The actual selection procedure is similar in some ways to the method developed by Naylor (1983) and adopted by Muller (1986) as described in Chapter 3, but in this case it is only used for the information selection process, not for all aspects of map design. It also has greater flexibility than Muller's method in that one only needs to change the method of deriving the selection index, to effect a change in the system's operation, whereas in Muller's system every input and output category would have to be re-evaluated.

Once the datasets have been selected a list will be displayed for the user's approval. It is likely however, particularly for more complex maps, that some re-evaluation of the information selected may have to take place after initial attempts at assigning representation methods.

RULES

1 Calculate selection_index

2 For each class of base data IF selection score >= selection index THEN add to selected list

3 IF topic includes a single theme data set THEN select this ELSE IF topic multiple possible themes ASK user to select those he wants (MENU)

4 CONFIRM selected information with user

RULES FOR GENERAL CARTOGRAPHIC REPRESENTATIONS

Although an experienced cartographer could devise an infinite number of symbols, for the purposes of developing an expert system this range must be somehow limited. Based upon the regime of cartographic representations developed earlier (Figure 5.2) the various major possibilities are discussed and guidelines presented. These are generally limited to the maps being considered in this study, in the scale range 1:2M to 1:15M, although many could be extended to other scales.

Further restrictions are placed on the symbols used by the limitations of the hardware and software used to develop the system. These are discussed in more detail in Chapters 7 & 8, but some general points are noted here. Point symbols are limited to a small number of simple geometric symbols. Line gauge is limited to normal (single pixel wide) and thick (3 pixel wide) lines. It would be desirable to have at least 3 line gauges available, and this is indicated in some rules by referring to fine, medium, wide and very wide lines. The actual gauge of the line used is less important than the contrast effects achieved. Once the system moves beyond the prototype stage a more sophisticated graphical interface would allow the more comprehensive range of point symbols and line gauges to be used.

There are many ways of describing colour, the detailed discussion of which is beyond the scope of this exercise. To keep colour descriptions simple, the hues Magenta, Red, Orange, Yellow, Green, Cyan, Blue, Purple and Brown will be referred to, plus Black, White and Grey. Adjectives such as pale, light, medium and dark are also used to indicate the lightness of colours. For more detailed specification percentage combinations of the subtractive primaries are preferred in cartography, although for computer graphic displays the image is generated by mixing proportions of the additive colours. For untrained users it is simpler to refer to the hues listed above and illustrate the range available on the screen.

The graphic displays used in the development of the system have a relatively limited range of possible colours. For this reason, and more general reasons of clarity, single hue look up tables are limited to a maximum of five lightness levels.

The rules for the precise definition of individual symbols make extensive use of look up tables (LUTs) which store default descriptions of various options for symbolisation. This includes point symbol collections, line styles and colour set specifications.

As far as possible symbolisation will be dealt with automatically by the system. The problem arises when additional information is required from the system user who may not be familiar with the cartographic terms used to describe the various representations (which not even cartographers agree on) or terms for the graphic variables. The most appropriate method of dealing with this in an expert system is to incorporate extensive explanation and help facilities and a dictionary of synonyms. For example, for each of the representation methods there should be a simple graphical example available which can be quickly displayed on the screen. Similarly, graphic representations of each of the look up tables should be able to be viewed. How these are described or titled verbally will require careful consideration.

Point Symbols

Generally these represent features or information occurring at points, or considered to be points at map scale. In some cases data may be collected for areas and the value assigned to a point within the area. The graphic variables used are point form; point size; and point colour. Point form refers to all variations in the shape of the symbol, including internal variations, additions, etc. Point size is self explanatory, although it only has a strict, accurate relationship with the feature when the scale or feature is large enough to show the true plan extent: in all other cases variation in size are related to other characteristics of the feature. Point colour refers mainly to variations in hue unless the symbol is large enough to cover a considerable area, in which case internal variations similar to those used for area symbols are possible.

0A **Repeated point symbols.** All points have same value.
a represents occurrences of discrete individuals or features
b represents some fixed quantity grouped for each symbol

This representation method shows the distribution of a phenomenon by means of a series of uniform symbols, frequently dots.

In its simplest form (**0Aa**) there is one dot per occurrence of the phenomenon, with locations known, giving arguably a version of 0B, the difference being that 0B represents multiple categories. An example of 0Aa could be the location of all power stations. Differentiating power stations by energy source would be 0B.

Frequently, each dot represents a number of occurrences **(0Ab).** Normally the data used is of the census type, with the number of occurrences within a given zone known rather than the actual location of individuals. Simply spacing the appropriate number of dots evenly (or randomly as done by several computer mapping packages) within the zone does not express the true distribution of the phenomenon, unless both the dots and the zones used are very small and the impression of a continuous variation in density is created (see 3A).

The advantage of representation 0Ab over area symbolisation (typically 2D - choropleth) is that when there is some information available about the likely distribution of the phenomenon within the zone a good representation of the distribution can be achieved. For example, in mapping population based on residency, topographic maps or aerial photographs can be used to eliminate from consideration areas which are obviously unpopulated, allowing the dots to cluster in areas which obviously are populated. Because of the need for additional information to reasonably locate the dots, currently this method is not a good candidate for automation. Consideration of its use should perhaps be notified to the system user, and guidance could be given on supplying the required additional information.

One common and very appropriate application of this method is the representation of rural population by dots in combination with urban population represented by graduated circles (0C).

FACTORS

dot shape - generally, round dots are used, although if the dots are relatively large squares, triangles, etc., can be used, or even pictographic symbols. The system is limited to true small round dots.

dot size - generally the dots should be small as smaller dots allow more detailed representation, though very small isolated dots are likely to be imperceptible. Mainly due to hardware limitations 3 fixed sizes of dots are available, although others could be specified by the user.

dot colour - This is limited to colours with high contrast against the background, as colour perception is limited for small symbols. Traditionally, the use of black or other 'dark' colours predominates. On a monitor however, it can be effective to use light dots on a dark background. Care must be taken with multi-colour backgrounds to ensure sufficient contrast is maintained against all the background colours.

dot value - ratio of occurrences per dot (0Ab only). Obviously the more occurrences per dot, the fewer dots there are and the less detailed the representation. Generally this ratio is found by trial and error, but a nomogram was developed by Mackay (Robinson et al., 1984; 304) which, although still requiring discretion, could be used as a basis for automatic selection. The interaction between dot size and value represents the major compromise between detail and visual effect in this type of representation.

multiple distributions - These may be depicted by varying the colour (normally hue) of the dots. It is necessary to ensure that dots do not overlap.

RULES - 0A

1* dots assumed to be round

2* 3 sizes of dot are available. Actual sizes will depend on resolution of output device. DEFAULT is medium size.

3 dots may touch, but should not overlap - affects dot size and dot ratio

4 IF points relatively clustered THEN use smaller dot size ELSE IF points relatively sparse THEN use larger dot size

5 use colour with high contrast against background

6 most suited for representing main theme information

7 can be combined with graduated symbols (0C) but avoid combinations with other point symbols.

0Aa

8 IF multiple distributions (i.e. more than one topic is to be shown by this method) THEN use rules for 0B

0Ab

9 use rules based on nomograph (Mackay/Robinson) to calculate dot value

10* maximum of 2 distributions allowed

11* IF multiple distributions THEN use large dots

12 IF multiple distributions to be represented THEN use total number in calculating density

13 IF multiple distributions THEN use high colour contrast between dots

0B **Categorised point symbols.** All features depicted by symbols of visually equal importance. Data feature coded.

A different symbol is used to represent each category of information. An example might be the representation of industrial plants. Each category would have its own shape or colour of symbol. If there is only a single category then the treatment is as 0Aa (e.g. all power stations represented the same would be 0Aa, differentiating coal, hydro and nuclear would be 0B). Each point in the database must be feature coded or have an appropriate attribute in the database. Symbols are assigned according to feature code or attribute value by use of a look-up table of symbols, either specifically for the phenomenon (preferred), or a general look-up table containing geometric symbols. Rules could be developed to select a range of appropriate geometric symbols indicating the relationship or hierarchy of features.

Symbol size will generally be small, typically 1 to 3 millimetres. For some distribution maps it may be desirable to use pictographic symbols, but these normally must be larger (3 to 5 mm.) and it may not be practical to depict these satisfactorily on CRT displays or with dot matrix printers, without increasing the size still further (Morrison & Forrest, in press).

As differences between classes are nominal, form and/or hue are the primary distinguishing variables. Sub-classes are often depicted by minor variation in form, although retaining form and changing colour, or retaining colour and changing form or orientation are also valid.

Perhaps the biggest difficulty with this type of representation is the likelihood of symbols overlapping excessively and hence becoming illegible. It is beyond the scope of the system developed here to check and resolve such overlap. A rule could be added that if overlap is extensive then the symbol size is too large or the scale is too small. In the longer term, the rules and procedures for determining and resolving such spatial conflicts devised by Mackaness (1986) could be incorporated.

FACTORS

symbol size - all should appear approximately the same size.

symbol form - selecting most suitable symbol; may be pictographic or geometric (many 'standard' or conventional symbols exist - use LUTs)

symbol colour - colour choice is usually limited, but very important. Because of the small sizes involved, variation in hue is the most important.

RULES - 0B

1 each category must be given a different shape of symbol AND/OR different hue

2* IF displayed with multi-coloured area representation THEN use variation in shape

3 use colours with high contrast against background AND to each other

4* do not use more than 12 categories (if possible). (Affects use of sub-classes)

5 IF small number of categories (<= 4) symbols can be small (2 mm), ELSE must be large (4 mm) to allow greater variation to be visible.

6* Hierarchy not allowed in monochrome

7 IF hierarchy THEN use colour variation for main categories with shape (square, circle, triangle, etc.) for sub-classes. (User may reverse this)

8 IF there are conventional or pictographic symbols refer to LUT. (Existence of this will be known from the metadata about the class of feature.)

- IF overlap extensive (define) THEN reduce symbol size OR increase scale. (Such checking is currently not included in the system due to the computational overhead involved.)

0C Ranked (hierarchical) point symbols.

These symbols may represent point locations, or areas considered to be points at map scale. A hierarchy should be immediately obvious from the representations used. The phenomenon most frequently depicted by this method is settlements ranked either by population or administrative status. These are not only shown on topographic maps but frequently on other special topic maps as part of the base information. In these instances the symbols are normally relatively small. It may be possible to use numerical values to derive ranks, but frequently this is not the case, e.g. the use of administrative status in classifying settlements.

FACTORS

symbol shape - normally simple geometric symbols are used, typically circles, squares, or a combination (e.g. use circles for low ranking and larger squares for high ranking). Symbols may be solid or outline only.

symbol size - generally relatively small symbols are used (0.5 to 2 mm).

symbol colour - usually all are same colour (typically black on printed maps). Due to small size high contrast is required.

RULES - 0C

1* use default LUTs in knowledge base to determine symbol sizes and shapes for 2, 3, 4, and 5 ranks OR give user choice of using circles, squares, or mixture, and solid or outline.

2 use colour with high contrast against background [black, red, purple] (assumes white or light background)

0D **Proportional or graduated point symbols.** Some visual impression of magnitude is given.

a graduated (classed) unipolar distribution

b proportional (unclassed) unipolar distribution

c graduated bipolar distributions [not considered]

d proportional bipolar distributions [not considered]

These point symbols may represent values relating to points; small areas treated as points at map scale; or values relating to areas, but represented at a point, normally at the centre of the zone. The data may be assigned ordinal classes (small town, big town, city, etc.), in which case the rules for representation 0C may be used. More frequently some value will be known which can be depicted on a continuous scale with symbol size varying in relation to the numerical value (0Db), or from which classes may be determined, using criteria similar to 2D (0Da).

If one of the primary purposes of the map is to illustrate a phenomenon by this method then the range of symbol sizes may be relatively large. A great deal has been published about the depiction of graduated symbols, but the important point is that if the data is grouped (0Da), there should be a perceptual difference between the size of symbol for each class, and a suitable legend. The number of classes will probably range from 3 to 7, more than this making the map user's task difficult. It is also common to use unclassed data (0Db), in which case the size of the symbol relates directly to the value. The perceptual issues involved in this latter form of representation are complex and producing satisfactory results with the wide range of values often involved can be very difficult. Legend design also presents problems. At this stage the system will only handle classed representations, i.e. graduated and not proportional symbols.

A certain amount of overlap of symbols is usually acceptable, although excessive overlap, particularly of similarly sized symbols, makes evaluation difficult. As a general rule large symbols are interrupted by smaller overlapping ones.

Mackaness' (1986) rules could be used to adjust symbol positions to reduce conflicts.

Symbols can vary in one dimension (length or height of bars), two dimensions (area) or simulation of three dimensions (appear to vary in volume). Only the second of these is considered at this stage. The length, area or volume respectively varies with the quantity being represented.

FACTORS

symbol form - although many shapes can be used, circles and squares are the most common, and discussion here is limited to these.

symbol size - maximum and minimum sizes of symbol must be determined. If symbols are too large they will overlap excessively. It is harder for the user to perceive variation if sizes are too small.

continuous or classified ranges - while continuous variation may be seen as desirable, if the ratio between largest and smallest value is large then there is difficulty in creating a scale without either the largest symbol being too large or the smallest symbol too small. Only classified ranges will be used here.

method of scaling - applies to continuous symbols where size is directly related to value. Various scaling methods can be used e.g. radius of circle or side of square, area scaling and perceptual modifications of these.

number of classes - too few classes reduce useful information; too many create confusion

class intervals - see 2D

filled or open - symbols can be in outline or can have the internal area filled with colour. If the symbols are large enough the fill may be constant for all symbols, vary in lightness to enhance the effects of size, or vary in hue to allow secondary classification of phenomena. For example, graduated circles of power station output may be colour coded to indicate generating method (coal, hydro, nuclear, etc.)

RULES - 0D

1* default symbol is circle. User may select square.

2* initial maximum size = 1/2 average distance between nearest neighbours OR set by system user.

3 IF data being represented is primary topic THEN solid symbols, ELSE outline.

4 IF outline only THEN use high contrast colour.

5 IF solid THEN use medium to high contrast against background.

6 IF secondary coding data available THEN use hue to differentiate categories.

7* Maximum of 5 secondary categories, colours by look-up table or by user

8* Multiple distributions not allowed by other means.

0Da

9* default is 5 classes. User may select 3 - 8

10* default is equal interval classes initially. User may modify this as for 2D classes (initially either quantiles or user specified).

11* use look-up table to determine symbol sizes, or user specify

0Db

- Scale symbols by area, i.e. use square root for computing radius/side.
- IF continuous scaling would give smallest symbol radius/side less than 2mm OR less than one tenth largest THEN use classified ranges (0Da).

0E Subdivided and multivariate point symbols.

a Fixed size

b Graduated

c Proportional

May represent information about points, areas considered as points at map scale, or areas (usually by zone centroid). Pie symbols most common but many possibilities. Simple rules as for 0D but many variations possible.

[not considered further]

0F Spot Values. Series of point locations with values. May have regular or irregular distribution. (e.g. Digital Terrain or Surface Model)

a sparse data e.g. spot heights

b dense irregular data e.g. triangular irregular network

c grid digital surface model

The simplest representation is a small dot with the value written beside it. Spot heights (0Fa) often used to supplement contours or show data points used to determine isolines. Soundings on hydrographic charts vary between 0Fa and 0Fb, but would generally be considered as the latter.

[not considered further]

Line Symbols

Generally these represent linear features or change in class of a surface, i.e. the outline of an area. They can also be used to represent movement, but this is not considered.

The graphic variables used are line form, line gauge and line colour. Line form refers to the structure of the line which may be continuous or broken, have additional elements, or be composed of multiple lines. Line gauge refers to the width of the line which generally depends upon relative importance of the phenomenon. This also includes the spacing of multiple lines. Line colour is normally limited to variations in hue unless the line is wide enough to allow variations similar to those for area symbols. Wide lines may or may not be enclosed by contrasting casing.

Maps for the screen present special problems for selecting line gauge, due to screen resolution and dot pitch. It is probable that gauge will be specified in multiples of dot widths (typically 0.25 - 0.4 mm) and may not relate directly to hardcopy values. The terms fine, medium, wide and very wide are used to refer ordinally to the gauge of lines. What these actually translate to will depend upon the graphics processor card and the monitor, or the hard copy output device. (The actual system is limited to single pixel and 3 pixel wide lines.)

1A **Boundary symbols.** Can represent:

natural boundaries (often zone of change)
actual ground feature forming boundary (often man made)
intangible (no real existence on the ground)

a nominal - all of same value - may be feature coded.

b hierarchical - typical of political/administrative boundaries.

Although a bounding line will be required for their production, the zone outlines of area symbolism methods may or may not be enhanced by line symbolisation. In some cases the change in area symbolisation may be all that is required, although this could result in the apparent amalgamation of some adjacent zones which may not be desired. The general rule is that boundaries of areas with contrasting area symbols can be fine lines. Boundary lines unsupported by area contrast need to be stronger.

How dominant the boundaries should be depends on the phenomenon and how it has been classified. If the boundaries in reality can be clearly delineated and there is a sudden change from one class to another over the boundary then the boundary may be depicted strongly. Frequently, however, boundaries are either approximate or an attempt to delineate one class gradually becoming another, i.e. the boundary is indistinct. In these cases the boundary should be less dominant graphically.

Administrative boundaries are commonly used as a convenient way of parcelling land to collect data, but the boundaries may exhibit little sympathy with the distribution of the phenomenon. Despite this, administrative boundaries often seem to be given a high level of importance on maps and are normally symbolised quite prominently. Indeed, they are frequently included as line symbols without the accompanying area symbolisation, and are probably the only boundaries sensibly treated in this manner.

FACTORS

line form - may relate to continuity or permanence of feature.

line gauge - relates to importance or permanence of feature.

importance - is specific linear feature required or is line required for change of area symbols or can it be implied by this?

coincidence - does the boundary coincide with other linear features? If so generally the other feature will be symbolised, or the boundary superimposed on the other feature.

RULES - 1A

1 use continuous line form unless rules for phenomena specify otherwise, e.g. use of LUT for administrative boundaries.

2 use minimum line gauge available unless boundaries main theme of map when LUT used. May be user specified.

3 IF boundary important (ASK user) THEN use colour with high contrast against background ELSE use low contrast.

4 is boundary symbol necessary or can it be implied by change in area symbol without outline? Use rules for phenomena or ASK user.

[5 IF boundaries are hierarchical THEN map purpose, level of detail and map scale can select levels to show OR user specify. (Part of select module?)]

1B Networks.

a feature coded, often hierarchical networks e.g. roads.

b branching hierarchical networks e.g. river systems.

Typically network data will be feature coded in a hierarchical scheme for representation type a), or a hierarchy is explicit or can be computed for type b). Symbolisation will follow this hierarchy for most data. In colour each network would be assigned a different hue, with sub-categories within a network being differentiated by line width or form, the exception being roads where several colours are often used, although this is less common on small scale maps. In monochrome the number of networks may have to be limited to avoid confusion. Hardware limitations may be important as it is often only possible to have a limited number of variations in line weight and form.

Although multiple lines and variation of casing and fill are frequently used at larger scales, at the very small scales considered here these are less common.

FACTORS

line form - generally at this scale continuous lines are used. Unless otherwise specified in a look-up table for the phenomenon, plain lines will be used.

line gauge - the greater the importance/significance the wider the line. Lowest level in hierarchy will generally get thinnest line possible and increase from that.

line colour - due to the use of thin lines, the main distinction will be in hue. Hues used need high contrast with background.

level in hierarchy - the number of levels which can be shown is dependant on scale, level of detail, etc.

RULES - 1B

1 use continuous line form unless LUT available

2 IF other lines (1A, 1B) THEN use different colour for each network OR different form.

3 IF network hierarchical THEN map purpose, level of detail and map scale can select level to show OR user specify

4 IF single level THEN use importance of feature (defined in knowledge base or by user) to determine gauge, ELSE IF hierarchy THEN use thinnest line for lowest level and increase up hierarchy. Ideally a lookup table for 2, 3, or 4 levels should be available. (For hard copy output a minimum difference of 0.1mm could be used) Some data sets will have specific look-up tables (e.g. roads). As only two line gauges are available for the screen, rules for the datasets will match the data to line width.

1C **Isolines (contours).** Lines representing a value and dividing the surface into zones between two values.

a uniform interval

b progressive interval [not considered, but see 2E]

The representation of lines of equal value should in theory indicate the nature of the data used to derive it. The data may be measured along the lines, such as photogrammetric contours, or derived by interpolation from a number of points. In the latter case '...its quality will depend largely on the density and distribution of the points and any assumption adopted about the gradients between them' (Keates, 1973; 62). Thus, measured lines are appropriately symbolised as solid lines, whereas lines interpolated from widely scattered points should indicate their lower precision by being symbolised with pecked lines. In practice this is rarely the case and solid lines are the most common. In addition, at the small scales being considered here, the lines are likely to be derived from larger scale sources and shown in a highly generalised form. Reliability is probably more important than high positional accuracy.

If a colour series is used for the phenomenon (such as hypsometric or layer colours) the isolines may either be implied by the change in area colour, or the lines can be made a more dominant part of the image. The latter is more likely to be the case for relatively simple special topic maps, where the primary phenomenon is being depicted by isolines and colour series, whereas the former approach is more typical of contours on topographic maps or where several phenomena are being depicted on the one map. The isoline interval may be smaller than that between area classes, with several lines within a layer.

While topographic contours typically have equal intervals, many other isolines use a variety of unequal intervals which often appear arbitrary, although usually there is a progressive, if uneven, increase or decrease in the interval. It is beyond the scope of the present exercise to attempt to rationalise this. The data made available to the system will be labelled with values. The system, based on the map scale may suggest reducing the total number of isolines by increasing the interval, but no attempt will be made to interpolate new isolines to force a particular interval.

FACTORS

line form - can be used to indicate accuracy/reliability (e.g. solid = measured; dashed = interpolated), but most often continuous plain lines are used.

line gauge - bold if important to theme of map or not supported by area colours.

line colour - contrast with background will depend upon importance. Normally all isolines will be same colour.

vertical intervals - contours are normally equal, others may be progressive or occasionally irregular. This will be incorporated into the data base. System abilities limited to selecting every other line for simplified output at smaller scales, or allowing user to select intervals.

index lines - these are emphasised lines every 4 or 5 times the standard vertical interval, using a convenient round number. Normally only applied to contours on topographic maps at scales >= 1:1M

multiple sets - it is quite common to have more than one set of isolines, e.g. average temperature and average rainfall. These are normally distinguished by colour, form, or both.

RULES - 1C

1 IF main theme of map AND no layer colours THEN use bold continuous lines.

2 IF layer colours being used AND lines required THEN use fine lines with medium contrast to layer colours. [black, white, grey, or same as base hue used for layer colours]

3 IF background information, use fine lines with medium contrast.

4 IF multiple sets, vary by colour OR form

1D Flow lines.

[not considered further]

1E Linear Cartograms

[not considered further]

1F Unstructured line symbols. Miscellaneous, non network, often isolated and/or discontinuous line symbols not included in 1A or 1B (e.g. fences, walls, geological faults, etc.)

Although widespread as part of the line image on large scale maps, features such as fences and walls are rarely depicted on small scale maps. Often they form boundaries and may be considered in part with 1Aa. Features such as pipelines, power lines, etc., can be treated as for networks (1Ba).

[not considered further]

Area Symbols

- each zone homogeneous; changes occur at zone boundaries.

The use of the graphic variables for area symbols is much more flexible than for either point or line symbols. Although the basic variables remain form, dimension and colour, more precise definitions of the major variations are useful:

Area form - the actual shape of the areas cannot vary (except for cartograms), although many authors (e.g. Robinson, et al., 1984) consider area patterns (see below) to be a variation in form for area. Here, area pattern is treated separately.

Area size - the size of areas cannot vary (except for cartograms), although again many text books refer to changing the size of pattern components to be a change in area size. This is treated here under area lightness and area pattern.

Area colour - full use of hue, lightness and saturation is possible for areas. Great contrasts in colour should be avoided if not warranted by the data: i.e. small variation in data value = small variation in colour.

Area hue - used mainly for variations in type or kind. In selecting hues, consideration must be given to their emotive effect. In representing natural phenomena such as precipitation and temperature colour associations are strong: e.g. red = hot and blue = cold.

Area lightness (value) - changes in lightness are used mainly to depict variations in quantity or importance. Graphically this includes variations in the lightness of solid hues as well as variation in the dimension and spacing of repeated point or line symbols either in the form of tints or visible patterns.

Area saturation - seldom deliberately used as part of classification system. Varying saturation can be used to create or enhance ordering or to help create visual balance. Its systematic use is beyond the scope of the proposed system.

Area pattern - Area patterns can vary widely in form (of component symbol(s)), orientation, dimension (both of symbol and spacing), and colour. Variations in pattern, signify nominal differences, but can be used to enhance Area lightness and/or hue differences. Although patterns can be assigned a lightness value, the visible structure of patterns allows them to be differentiated from areas of continuous colour, either solid or tint. The term area colour will be used to refer to the latter type of representation.

Apart from 2A, all area representations depict phenomena as continuously covering the surface. The symbolisation will generally cover the whole area of interest. All areas are delimited by (boundary) lines. Change in symbolisation can only occur at these boundaries. The boundary may be specifically symbolised or implied by change in area symbol. This will depend upon the nature of the boundary and will be determined by rules for the phenomena.

2A **Isolated areas.** Binary (dichotomous) division of surface - special case of 2C.

The simple subdivision of the surface into areas which do have some particular characteristics and those which do not is a common occurrence in mapping. For example lakes; woodland areas; oilfields. The surface distinctions show actual area of occupancy of the phenomenon and are usually nominal. The symbolisation of this type is graphically simple, the subject areas being symbolised and the remainder left blank, creating a distribution map.

Other candidates for this classification would be 'non study area', such as countries adjoining the country of study, which are often depicted in a pale grey tone, or the sea. Often this type of symbol is a major component in creating a visual hierarchy on the map.

This is probably the only type of area symbolisation that can coexist with other area symbols satisfactorily, without very complex rules of interaction, but can affect the choice of either continuous area colour, or pattern.

FACTORS

Colour or pattern? Simply, if the areas do not overlap other area distributions, or take precedence over them (typical of water areas) then continuous colour used. If there is overlap then use pattern, e.g. land over 1,000 metres on layer coloured temperature map.

RULES - 2A

1 IF areas do not overlap other area symbolisation or precedence exists (rules for phenomenon) then use Area Colour, ELSE use Area Pattern

2 IF phenomenon is part of main theme THEN use medium to high contrast colour or pattern ELSE use low contrast colour or pattern

3 IF areas small at map scale THEN use solid high saturation colours

4 IF other area symbols being shown THEN use medium to high contrast with those.

2B Unclassed. e.g. 'Political' map.

a single level

b hierarchical e.g. Country/state/county

In theory each area should be given a unique symbol. In practice, the main rule is that each area must be assigned a symbol (colour or pattern) different from each of its neighbours, but appearing to be of equal importance by controlling lightness and saturation (**2Ba**). A minimum of four colours is required for this, although more may be used. With the hardware limitations imposed on the system it would be difficult to obtain more than 8 or perhaps 16 colours, or 8 appropriate patterns that are distinctly different, but all appearing to be of equal lightness. For the purposes here the top level of the hierarchy will use up to six colours or patterns in look-up tables.

One solution to assigning symbols is to have each zone assigned a feature code for its symbolisation in the database. This would, however, restrict the possibilities of symbolisation. It is possible, given the relationships of the zones (i.e. the topological structure), to automatically assign symbols to the zones to meet the above criteria. This is a more flexible approach as it allows the number of colours or patterns to be used to be determined when the map is being designed, not when the data is being entered into the database. No implementation is known of this in existing mapping systems.

If there is a hierarchy of zones (**2Bb**), the problem is a little more complex. At the top of the hierarchy the solution is the same as above. Lower down the hierarchy there should be an association between sub-zones within a zone which is greater than that between zones. One solution to this is to distinguish zones by hue and sub-zones by lightness, although the number and contrast of sub-zones will depend upon the zone hue. While this use of lightness ostensibly breaks the fundamental rule of this type of map, as long as the values used and their distribution do not appear to create a graded scale and that the purpose of the map is made clear, this scheme can be used to good effect, e.g., the Political maps in Collins Atlas of the World (1983). Varying saturation as well as lightness can help to retain visual balance between the sub-zones. A more technically correct solution may be to depict sub-zones in the zone colour and use variation in pattern to distinguish them.

Again, due to hardware limitations, two levels of hierarchy would be the maximum that could be depicted adequately. In colour, level 1 would be distinguished by hue, with level 2 by lightness or pattern. In monochrome it is probably only realistic to use area symbols for one level, i.e. patterns, but it would be possible to use texture to create a second level e.g. a diagonal line pattern may be modified by changing the spacing and width of the lines, but retain the same 'lightness' (Bertin's grain), or perhaps changing the orientation, but this can lead to displeasing visual effects and is best avoided.

In selecting the colours or patterns for zones, an attempt can be made either to maximise or minimise the differences, e.g. one could choose red, blue, green and yellow, or one could choose four colours in the blue/green area of the spectrum. Several recent atlases show a tendency towards the use of less saturated colours than was once the case. This choice can probably only be made by the author with examples of the possibilities being displayed.

FACTORS

area colour assignment - arbitrary assignment of colour (hue), but no adjacent areas have same colour (minimum of 4 colours required).

area colour - choice of colours/colour range

- for uniform all should appear equivalent.
- for hierarchical - for all sub-classes, variation within a class should appear less than between class variations. e.g. classes in red, blue, green, etc. Sub-classes in shades of red, blue, green, etc., pattern, or texture.

level of hierarchy to display - Practically limited to two levels of hierarchy.

RULES - 2B

1 No adjacent zone may have the same symbol

2* Use scale and level of detail to determine level of hierarchy to show OR ask user

3* Default is 5 colours, use look up table of equivalent colours OR user can specify up to 6 colours or patterns.

4 IF hierarchy to be shown, use hue for first level and lightness (4 tints) or patterns (4 or 5) for second

5 Cannot be combined with other area symbolism except [2A]

6 ASK user to select colour scheme from 3 options displayed (e.g. light colours, strong colours, etc.), or create own LUT

2C **Categorical.** Data feature coded. e.g. land use, vegetation, geology.

a uniform

b hierarchical (i.e. includes sub-classes)

This representation is used for a great number of phenomena, such as land use, soils, geology, vegetation, etc. A feature of these phenomena is that they tend to be classified into a relatively large number of classes and sub-classes. Also, in some cases there are standard area colours used for their depiction.

Like (2B) above, in theory each class should appear to be of equal importance and therefore of the same lightness value, thus the primary distinction between classes will be one of hue or pattern. Due to the number of classes and/or sub-classes this is not always possible, but variations in lightness should be used in a way which does not imply quantities or importance.

For this representation, feature coding must be included in the database and symbolisation will be assigned according to this. If there is a 'standard' symbology for a phenomenon, this may be included in the knowledge base. For user supplied data the map author will have to assign symbols to the top level of the hierarchy at least, although the system would select the appropriate number of colours or patterns. The system will then be able to symbolise sub-classes automatically if desired, provided that there is a limited number.

The solution to this type of symbolisation is basically that which Sampson and Poiker (1985) attempted to automate, as described in Chapter 3.

FACTORS

usually main theme of map

area colour - assigned via look-up table, either using defaults tables (as for 2B) or specifically for phenomenon.

level of hierarchy to display - depends upon scale and level of detail.

hierarchies - for all sub-classes, variation within a class should appear less than between class variations. e.g. classes in red, blue, green, etc. Sub-classes in shades of red, blue, green, etc., or use of pattern.

reclassification of attributes or combination of sub-classes may be required if too many feature codes or if scale required creates many very small areas.

RULES - 2C

1 use scale and level of detail to determine level of hierarchy to show.

2 IF average area of sub-class zones too small [define] THEN show top level only

3 If average area of zones too small [define] THEN suggest larger scale, OR let user redefine classes.

4 IF known use 'standard' symbolisation ELSE use rules for selecting required number of colours and patterns.

5 Cannot show other area symbols except [2A]

2D **Graded series.** Two possibilities for boundaries, but treatment of symbolisation the same:

> **Choropleth.** Discrete zones with numerical values - boundaries imposed on distribution (normally administrative, but could also be grid).
>
> **Dasymetric.** Discrete zones with numerical values - boundaries derived from distribution. (Could be from fine grid with internal boundaries removed.)

a unipolar distribution

b bipolar distribution

c bivariate (not Dasymetric) [not considered further]

The representation scale of the phenomenon may be single or double ended (bipolar). For a single ended scale, variation in value (lightness) is used, although how this is applied depends on the phenomenon. Normally the use of colour in such scales is metaphysical, i.e. the stronger (darker) the colour, the greater the quantity. For bi-polar scales the lightest colour would be used for the middle value with progressively darker colours away from the this, e.g. for population change, yellow or white for little or none; progressively darker reds for increase; and progressively darker blues for decrease. Bivariate schemes involve representing two variables simultaneously in a cross-tabulated form.

The phenomena represented by this method can vary widely, although generally the data are quantities aggregated over areas. The most common data to be represented in this manner is census data which has been collected by enumeration districts, but aggregated for larger statistical areas, commonly the same as administrative units. It is arguable that populations are not suitable for mapping as a continuous phenomenon, and the choropleth technique has been much overused, but in the absence of a more detailed breakdown of data it is at least possible to show some measure of the distribution. The value of the method depends largely on the sizes of the zones and how well the boundaries relate to actual changes in the distribution. If data is available at the sub-zone level (e.g. unpopulated areas) then its use may help to remove some of the faults of this type of depiction.

The boundaries employed are not usually natural boundaries of the phenomenon but imposed political/administrative units. Due to variation in the size and shape of the zones it is rarely satisfactory (despite common practice) to represent total values. Either densities or ratios are more appropriate, or zonal averages for the phenomenon. If the zones are of similar size (e.g. grid square data) then absolute values may be appropriate.

Although it is possible to produce an 'unclassed' map by assigning a grey scale value (lightness) to each zone in proportion to its value (in effect having as many classes as there are zones), typically the zones are grouped into a number of classes based on their data value. Commonly 4 to 7 classes are used, but most choropleth mapping programs allow up to ten classes (some allow many more, but this is generally pointless). Too many classes are undesirable as the number of perceptible differences is limited and it becomes difficult for the map user to distinguish between them. Experience indicates that 4 or 5 classes are probably a good basis from which to start examining the possibilities.

In addition to choosing the number of classes, the boundaries between classes must be selected. There are numerous methods which may be employed, common ones being equal intervals and quantiles. Typically the values are not normally distributed so to try and more adequately match the distribution, some methods are based on a more complex statisitical classification of the data, or employ arithmetic or geometric progressions. Examination of the data values for 'natural breaks' is in many cases the best solution, but, unless some form of pattern matching can be applied, it is unlikely that this can be automated. What can be automated is the calculation of some measure of 'goodness of fit' of various class interval schemes to the data, and although this cannot give a definitive answer, it may be used as a guide.

The choice of the number of classes and the class intervals is a more complex and skilful task than is apparent from the widespread use (abuse?) of choropleth maps. Although often left to the cartographer, this matter should be resolved by the map author (in consultation). More familiarity with the phenomenon and the nature of the available data by the map authors would improve matters. This whole aspect could be developed into an independent expert system and detailed development is beyond the scope of the current project.

The choice of colour for this general class is quite wide, but some care must be exercised. While it is important to differentiate classes, large differences in lightness may imply larger differences between data values than is warranted. This is one of the few types of representation for which the perception of screen images has been studied (e.g. Dobson, 1984a; McGranahan 1985 & 1986a, b; Gilmartin, 1986; Gilmartin & Shelton, 1989) and some of the guidelines discussed are incorporated.

FACTORS

number of classes - usually the data values will be continuous. Unless a continuous representation is required the data must be divided into classes.

class intervals - the method of dividing the data into classes.

selection of base hue(s) - some common colour schemes exist.

determination of 'grey scale' - depends upon hue, number of classes and data range.

RULES - 2D

1* Default is 5 classes but allow user to specify 3 - 8.

2* Default is equal interval classes OPTIONS: Quantiles; User specified.

3 IF classes <= 5 THEN single hue set of tints - [Greens, Yellow/Browns, Reds, Magentas, Greys]
ELSE IF classes > 5 THEN part spectral range of tints [Yellow-Green, Yellow-Red, Magenta-Purple, Blue-Purple]

4 IF bipolar THEN user must specify classes and change over point.

5 IF bipolar THEN choose complementary colours: Reds-Blues; Reds-Greens

- USE rules of equal appearing steps for grey scale (look-up tables for each hue)

2E Layer coloured series.

a unipolar distribution (e.g. average rainfall)

b bipolar distribution (e.g. temperature zones)

Graphically this is the same as the previous representation, the difference being in the type of phenomena represented and hence the spatial relationships of the zones.

The structure of the information is quite different from those areas considered thus far, as here a continuous variation over the surface is implied rather than a sudden discontinuity at a boundary, although the symbolisation can only change at the boundary. This method is an enhancement of [1C], and uses the same data.

Many of the considerations discussed above (2D) apply, although it is easier for the user to perceive a larger number of classes due to the spatial arrangement (nesting) of zones. There are several conventions for phenomena normally depicted this way, such as the use of blues for rainfall, red for high and blue for low temperatures, etc.

Like [2Dc] above, the depiction may be bipolar with distinct ranges of colour used above and below a specific value, which may be zero, the average value, or some arbitrary value.

Data available will most often consist of isolines based on a relatively small number of point observations; therefore the portrayal of the phenomenon will be highly generalised. Selecting the number of classes and their range will seldom be a requirement, although it may be desirable to merge classes on smaller scale maps.

FACTORS

selection of vertical intervals - The data available will be the limiting factor, but some multiple of the isoline interval may be used.

selection of base hue(s) - most 'natural' phenomena have some associated hue(s). For others it is arbitrary.

determination of 'value scale' - depends upon hue and number of classes.

RULES - 2E

1 Use rules for phenomena to select base colour(s)

2 Use LUTs of colours and patterns

2Da

3* maximum classes = 8

4* IF scale very small or detail low THEN max. classes = 5

2Db

5 change point = 0 or user specify value

6* maximum intervals = 10 (5 above, 5 below change point)

2F Hypsometric and Bathymetric colours

Spatially the same as **2E** but difference in graphical treatment warrants separate class.

Normally a progressive interval is used for heights (and depths). A variety of colour schemes have been used for heights, although pseudo-spectral schemes such as greens/yellows/reddish browns/purple is currently popular. The only obvious solution to this is to have standard height ranges and use LUT.

Bathymetric colours are simpler as usually only blues or blues and white are used.

RULES -2F

1 Use LUT for colours

2G Area Cartograms. Areas scaled by a variable.

[not considered further]

SURFACES

- These differ from areas in that continuous variation is represented.

3A Shading. Symbolisation has continuous variation in lightness (e.g. hill shading).

In addition to hill shading, this form of representation has also been suggested for representing 'continuous' quantitative or statistical surfaces, by using semi random dots in a set of cells on raster output devices (e.g. Groop and Smith, 1982).

[not considered further]

3B Block Diagrams.

a continuous

b stepped

Arguably these are not true maps as they are non-orthogonal. Sometimes referred to as 2 1/2 D representations.

[not considered further]

3C 3D Surface Model.

It is not possible to display information in this form (except as a physical model or hologram), but there is increasing importance in this as an internal representation in a computer. The visualisation and rendering of the 3D surfaces could be developed in this class

[not considered further]

RULES FOR PHENOMENA IN DATABASE

Here the rules applying to the various phenomena included in the database are given. After the name the possible representation types are given (see above). Default values for the factors involved are given followed by a list of possible alternatives in square brackets. The majority of the rules presented here are system specific, although several may well extend to other scales or uses.

Coastline (1A) This will be included on all maps (assuming the area includes the coast).

Line_colour	blue [black, white]
Line_form	simple continuous
Line_gauge	fine (single pixel)

Seas (2A) Generally symbolised except for outline maps.

Area_colour	light blue [white, light grey, dark blue]

Rivers (1B)

Line_colour	blue [black, grey, white]
Line_form	continuous
Line_gauge	Levels 2 and 3 - finest lines; Level one 5:3 of level 2 & 3.

Inland Water Bodies (2A, 1Aa)

Normally these will be shown if rivers are shown.

IF Rivers shown THEN outline shown in River line_colour

Area_colour	light blue [dark blue, white, light grey] (normally same as Seas)

Political Boundaries (1A)

Level 1 (International) shown by default - may be deselected

Level 1 must be shown when countries selected

Level 2 must be shown if first level internal division shown
Level 3 must be shown if second level internal divisions shown
Line_form   level 1 long dash/short dash OR continuous
             level 2 long dash/short dash; level 3 short dash/dot
Line_gauge   level 1 medium, level 2 & 3 fine
Line_colour level 1 [black, red, grey]; level 2 & 3 [black or grey]

Countries (2Ba, 1Aa)
IF other area symbols (except 2A) THEN show by boundaries only
Default colour set = 5 hues, low saturation, medium lightness. Other LUTs available.

Internal Divisions (2Bb, 1Ab)
Political boundaries must be shown to at least same level in hierarchy
IF size of units small [define] THEN show level by boundaries only (1Ab) and omit lower levels
IF countries symbolised by 2Ba THEN first level divisions shown by tints and shades (5) of country hue, second level by patterns (5) or boundaries only
ELSE IF countries symbolised by 1Ab THEN first level use hues (default set as for countries) and second level tints or patterns

Settlements (administrative status) (0C)
LUT of default symbol form and size for 1, 2, or 3 levels
Colour - solid magenta [red, dark grey]

Major Urban Areas (2A)
Only show when scale > 1:2.5M
Area_colour - Grey [white, yellow, orange]

Roads (1Ba)
LUTs for 1, 2, or 3 levels being shown
Line_form   continuous
Line_colour Red [grey, yellow, black]. (note LUT may include dual coloured lines, e.g. thick red with thin yellow fill. Line_colour refers to the outline colour as contrast between this and the background is the most important consideration. Normally all road classes will have same primary colour, but may have different secondary colours.)
Check for conflict with other line symbols, especially boundaries

Line_gauge	Level 1 only -	medium (0.4mm)
	Level 1 & 2 -	Level 1 medium, level 2 fine (0.2mm)
	Level 1 2 & 3 -	Level 1 thick (0.6mm), level 2 medium, level 3 fine

Railways (1Ba)

Line_form	continuous
Line_colour	black [grey]
line_gauge	fine

Relief (2F, 1C, 2A)

IF number of levels = 1 (e.g. land over 1000m) then 2A
ELSE IF other area symbols (except 2A) then 1D (rare at these scales)

2F	specific LUTs for 2 to 8 levels	
1D	Line_form	continuous [dashed]
	Line_colour	brown [orange, black, white, grey]
	Line_gauge	fine
2A	IF other area symbols THEN Area_pattern = fine black [white] diagonal lines	
	ELSE grey tint	

Geology, Soils, Land Cover, Land Use (2Ca)

Specific LUTs for each, based on feature classes

Annual Precipitation (2Ea, 1C)

IF other area symbols (except 2A) then 1C

1C	Line_form	continuous [dashed, chained]
	Line_colour	blue [white grey black]
	Line_gauge	medium [fine .. thick]
2Ea	IF classes < 5 use blue LUT	
	IF classes >= 5 use yellow/blue LUT	

Ave Annual Temperature (2Eb, 2Ea, 1C)

IF other area symbols (except 2A) THEN use 1C
IF min temp < 0 THEN 2Eb, with 0 as change point
IF min temp >= 0 THEN 2Ea [user select 2Eb and change point]

1C	Line_form	continuous [dashed, chained]
	Line_colour	red [white, grey, black]
	Line_gauge	medium [fine .. thick]
2E	IF classes <= 5 use red [magenta] LUT	

IF classes > 5 use yellow/red LUT
2Eb Use blue LUT for classes with values < 0
Use red LUT for classes with values > 0

Population (2Da, 0Da)
IF population density required THEN compute and use 2Da
ELSE IF Total population required THEN use 0Da
2Da
Area_colour Greens (yellow/brown, reds, magentas, greys, part spectral)
Use colour LUT, depending on number of classes.
0Da
Point_colour red [magenta, orange, white, grey, black]
Point_size use LUT (ideally should compute size based upon number of points, map area and average spacing, but this is not implemented currently)

Urban Population (Settlements) 0Da
Point_colour red [magenta, orange, white, grey, black]
Point_size use LUT (ideally should compute size based upon number of points, map area and average spacing, but this is not implemented currently).

Rural Population (2Da)
Compute or use density
Area_colour Greens (yellow/brown, reds, magentas, greys, part spectral).
Use colour LUT, depending on number of classes.

Population Change (2Db)
IF even number of classes THEN compute classes above and below zero
ELSE have middle class centred on zero.
Use bipolar colour LUT, depending on number of classes.

CHAPTER SEVEN

Developing a prototype map design expert system

> Many problems studied within AI call for inference. Unfortunately, the general purpose inference mechanisms which have been built in AI are known to be susceptible to combinatorial explosions. One solution to this problem is to build domain-specific inference mechanisms. Building such a mechanism is non-trivial and, by definition, the researcher can inherit little from those which have been built before.[1]

THE SYSTEM

The MapDesigner system essentially consists of four parts. These are the database of geographic information; the knowledge base which holds information on how to use the data; a user interface; and a control program (inference engine) which interacts with the user interface, the database and the knowledge base and guides inference. In addition to these primary components, several supporting elements are also required for a complete cartographic design expert system. As shown in Figure 7.1, these include a knowledge acquisition system and an explanation system, required for all expert systems, together with a graphical output system, a geographical data input system and a geographical data description system required specifically for a cartographic expert system. The geographical descriptions, or metadata, could be incorporated into the main knowledge base, or stored independently as illustrated. Before detailing each of these components, some general points about the selection of hardware and software for developing the system should be considered, and the effects these decisions may have on the resulting system.

DEVELOPMENT ENVIRONMENT

Hardware

Although the best advice in 'going digital' is first to define your problem, find the software that can solve the problem and then buy the hardware which that

[1] Bundy, A., Byrd, L., Mellish, C.S. Special-purpose but domain independent, inference mechanisms. In: Steels & Campbell **Progress in Artificial Intelligence.** Chichester: Ellis Horwood Ltd., 1985, p.93.

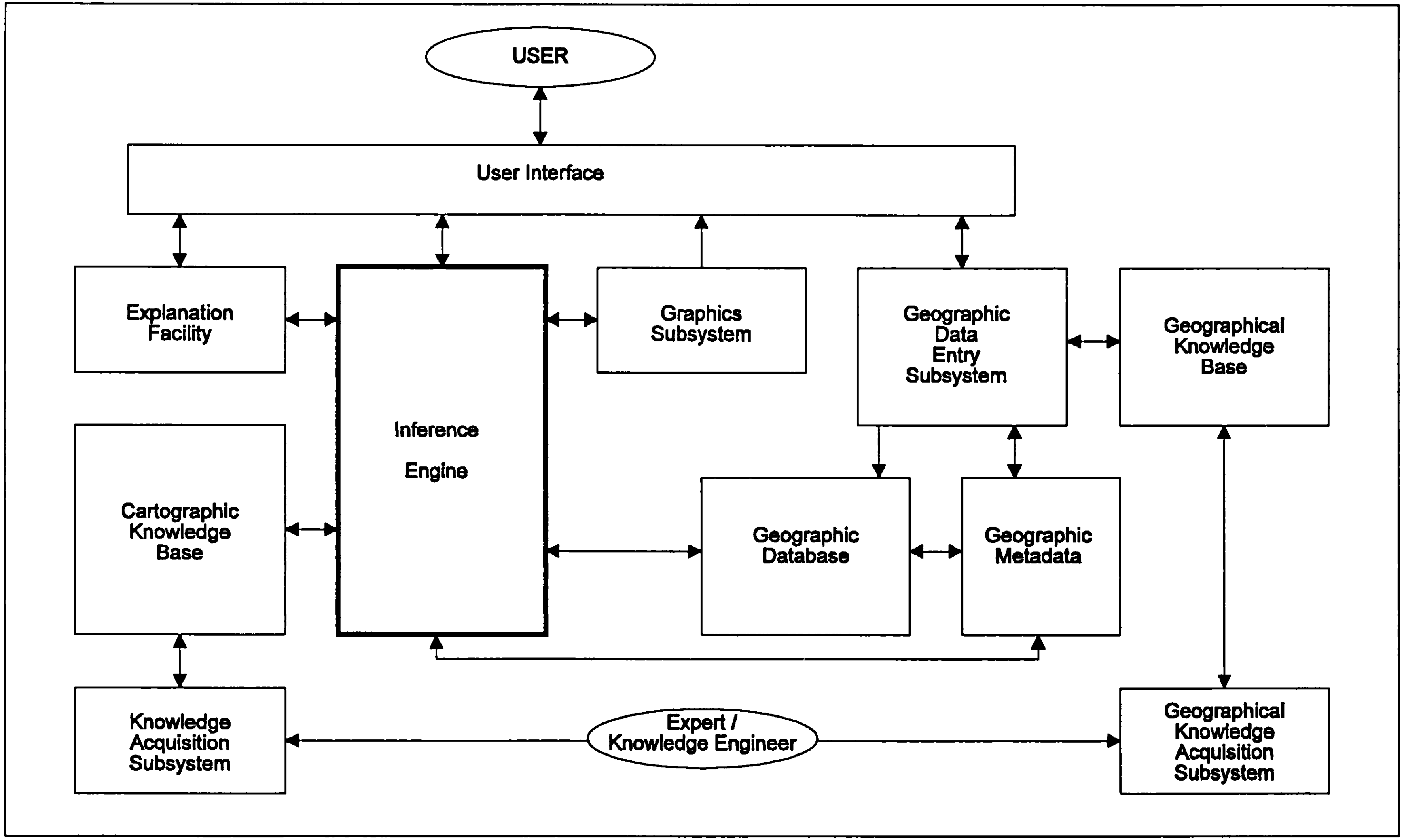

USER
User Interface
Explanation Facility
Inference Engine
Graphics Subsystem
Geographic Data Entry Subsystem
Geographical Knowledge Base
Cartographic Knowledge Base
Geographic Database
Geographic Metadata
Knowledge Acquisition Subsystem
Expert / Knowledge Engineer
Geographical Knowledge Acquisition Subsystem

Figure 7.1 The Architecture of a Cartographic Design Expert System

software runs on, in setting out to develop a system slightly different criteria come into force. Although ten years ago personal computers seldom had the necessary power, memory capacity, or graphics capability to serve as the main development platform for a comprehensive design expert system, the current generally available models certainly are sufficiently fast and powerful to develop a system to the prototype stage, if not the full production stage. Indeed, with the release of the Windows NT operating system there has been a considerable increase in the possibilities of comprehensive GIS packages moving onto the PC platform as witnessed by the current move by Intergraph from Unix workstations to top of the range PCs running Windows NT.

Perhaps one of the most significant developments of PC hardware from the mapping view point was the introduction of the VGA graphics standard in the late 1980s and its subsequent adoption as the virtually universal graphics standard by most systems. Although there are now enhanced version of the VGA standard, its ability to plot at a resolution of 640 by 480 pixels in 16 colours from a choice of 64 resulted in a practical system for plotting coloured maps without recourse to esoteric, specialist graphics adapters which would limit potential distribution of any system relying upon them.

Thus, the decision was made to develop the system based on fairly basic, standard hardware available. As developed, the system could conceivably run on an 8088 based PCXT with 640k RAM, but a more realistic minimum is an 80386 PC with 1Mb of RAM and a hard disk. A standard VGA graphics card is also required.

The system as developed is compiled and no additional special software or software licence (or hardware) is required to run it.

Software

There are two options in developing an expert system: use an existing inference engine, i.e. use a proprietary expert system shell; or develop an inference engine specifically for the system. An evaluation of several Mainframe and PC based expert system shells indicated that none had the facilities required to develop a cartographic design expert system. The main difficulties are in integrating the inference mechanism with the required cartographic database and the production of graphic displays. Some shells, such as VP Expert and Knowledge Pro do have the ability to incorporate graphics, but these are either pre-created stored images or simple charts and diagrams[2]. Robinson and Jackson (1985) examined the use of expert system shells for the MAPAID system and reached the conclusion that they were inadequate for the complexities of map design. Thus, at an early stage a decision was taken to develop a customised inference mechanism. This view has also received support by others developing map design systems (e.g. Muller and Wang, 1990; Zhen et al, 1993) and Intelligent CAD systems (e.g. Ditterich and Ullman, 1987).

As noted previously, knowledge engineering is frequently cited as being a major bottleneck in the development of expert systems. A customised inference mechanism has the advantage that the format of the knowledge base can be designed to facilitate the knowledge engineering process and the nature of the problem to be solved. Thus, the structures used to represent the knowledge can be as close as possible to those used by the domain expert (Merritt, 1989; 4), or at least those that offer the most appropriate representation. This further supports the development of a customised inference engine.

Having decided to develop a specialist inference engine, the choice of development language must be considered. While traditional (procedural) languages such as Pascal, C and Basic have been used for some expert system development, they generally are viewed as being less than ideal and most opt to use 'fifth generation' languages. These languages support the declarative programming approach (see below) which is more suited to the type of problem solving involved with AI and ES rather than the procedural approach adopted by more traditional languages.

Amongst those developing cartographic expert systems and systems in other fields, Prolog is one of the most popular languages. In this case Turbo Prolog by Borland International was adopted for a number of reasons. First, it was relatively inexpensive when first introduced (under £200). Borland had developed a reputation for good quality language products and their introduction of a Prolog compiler quickly led to a much wider distribution and knowledge of Prolog. Second, the author was familiar with other Borland products such as their Pascal compiler and the development environment was similar. Unlike most versions of Prolog for the PC, Turbo Prolog allowed the use of real arithmetic not just integer arithmetic, clearly essential for processing map co-ordinates, etc. The compiler includes a set

[2] The 1994 release of Knowledge Pro for Windows (Developers Edition) reportedly has the capability to plot the required graphics and merits further investigation.

of basic graphical commands which could be used to plot maps on the screen, a facility not included in other Prologs available at the time. And, finally, the system generates true compiled code which does not require an interpreter to be used at run time, resulting in significantly faster execution of the system.

The initial version of the compiler was quite limited in the scope of programs which could be developed, mainly due to memory limitations, but since Version 2 it has been possible to develop modular programs which overcome many of the limitations of the earlier version. In the early 1990s Borland withdrew the product but it has continued to be made available by PDC of Denmark, the original developers of the system for Borland. The current version is Version 3.3, although most of the changes since Version 2 are minor. There are in fact four different V3.3 compilers: a DOS compiler, an extended DOS compiler, a Windows version and a Unix compiler. As initial development took place in the Prolog V2 and V3.0 compilers under DOS, this environment has been retained. It was hoped with Version 3.3 to use the extended DOS compiler which allows access to memory beyond the DOS 640K limit, but the extended version does not allow graphics, which would entail writing a graphics system in C which could be linked to the Prolog system. The Windows compiler has not been investigated, but clearly there is potential for future developments in that direction, particularly as the graphical capabilities under Windows greatly exceed those currently available.

There have been some problems with the compiler, mainly due to memory restrictions. Prolog is notoriously memory hungry and the DOS memory limit and the necessary use of modular programming place some limitations on the flexibility of the inference engine. Also, the development environment is not very stable, with unexplained crashes of the system happening rather more frequently than would be desirable. On many occasions a version of the program which previously compiled satisfactorily would lock the whole computer on subsequent compilation, requiring a 'cold reboot' (i.e. switching the whole system off).

In addition to the compiler, use has also been made of the PDC Prolog Toolbox, a suite of routines which can be 'included' in programs being developed. Many of these are related to the user interface and extensive use has been made of the menu system. There is also a further development of the graphics system, the main advantage of which is a built-in scaling system to convert from an arbitrary co-ordinate system to that of the selected graphics device, and some use has been made of this.

PROLOG

Prolog was fist developed in the early 1970s at the University of Marseilles as a convenient tool for programming in logic (hence pro - log) and is now one of the best known languages for artificial intelligence development. The generally accepted standard definition of Prolog is the description of the language given by Clocksin and Mellish (1981).

Unlike conventional programs in languages like Pascal, in Prolog a description of the problem is given by a series of facts and rules and the program asked to find all solutions to the problem. The essential premise is that the programmer describes what must be done, but the Prolog system itself organises how the computation is carried out (Borland International, 1986). This is known as a 'declarative' approach, rather than the 'procedural' approach to programming adopted by Basic, Pascal, etc. where the programmer is also concerned with the mechanisms of solving the problem. This in theory should result in Prolog programs being shorter (i.e. requiring less lines of code) than procedural programs, easier to program, and easier to read and understand. While this may hold true for relatively simple programs, as program complexity increases programming in Prolog is not necessarily as simple of some of the introductory textbooks might imply. What is clear is that the thought processes involved in programming in Prolog are quite different from procedural programming and the conversion from programming in one paradigm to the other is not trivial.

PDC Prolog

Although conforming to the basic principles outlined by Clocksin and Mellish (1981) the implementation of the Prolog language in the PDC Prolog compiler has some distinct features, some of which extend the language and others which place restrictions on it. The discussion of Prolog here concentrates on this specific version, although some of the differences from 'standard' Prolog are mentioned.

Essentially a Prolog program is descriptive: it consists of a description of how to solve the problem. This description is made up of two basic components: a series of named *objects* involved in the problem; and a set of *facts* and *rules* describing relations between objects. A PDC Prolog program consists of a series of initial declarations defining objects and their structure followed by a list of logical statements combining the rules and facts. Most implementations of Prolog do not require the detailed declaration stage required by PDC Prolog. The system operates by trying to solve a *goal*. When the program is executed the system tries to find all possible solutions to that goal. If only one solution is required, execution can be stopped at that stage.

Objects

The PDC Prolog compiler is known as a 'typed' compiler. That is, before any variable is used it must first be declared and its characteristics defined. The first section of any program is the 'domains' section which contains these declarations. The four main standard (built in) domains are integer, real, string, and symbol. The first two are self explanatory. Strings are any series of alphanumeric characters enclosed in double quotes e.g. "this is a string". Symbols can have two formats: the first is identical to a string; the second is any sequence of letters, numbers or underscores (_) starting with a lower case letter and not including spaces, e.g. map_topic.

In addition to the standard domains, the programmer can also declare custom domains. These custom domains are always composed of a standard domain, a combination of the standard domains, or custom domains already declared. They may be a subset of an existing domain. A common example is the 'key' domain which includes definitions of all the keys on the normal computer keyboard. Domains such as 'row' and 'column' both declared as integers are useful for making other definitions easier to read and understand, etc., rather than simply using the standard domains. For referring to dates as a single entity a 'date' domain could be declared:

```
    date  =     date (Integer, Integer, Integer)
```

List domains, which consist of a list of terms all in the same domain can also be declared, e.g. Integerlist = a list of integers. Lists can be passed by the program as a single entity or the contents of the list can be worked on individually. Lists and operations on them are a very important aspect of programming in Prolog and are discussed in more detail later.

So, for example, we could declare a domain 'point' which contains the pair of real values indicating the position of a point and a domain 'line' which consists of the list of co-ordinate pairs making up the line:

```
  DOMAINS
    point =     point (REAL, REAL)
    line  =     point*
```

where the * indicates a list of the denoted domain. This would allow the whole set of co-ordinates describing a line to be passed as a single entity to a plotting routine for example.

Domains can be quite complex as becomes apparent from the 'symbolspec' and 'symbolspeclist' described in Chapter 8.

Facts and Rules, Predicates and Clauses

'Facts' and 'rules', referred to as 'terms', are used by Prolog to relate objects. These terms are defined by a 'predicate' declaration and used operationally as 'clauses'. Many versions of Prolog do not require predicates to be specifically declared before use, but PDC Prolog does. The declaration indicates the domain of each element in the predicate call, e.g.

```
predicates
  colour (symbol, symbol)
  level_of_detail (integer)
```

Thus, the predicate call for colour includes two objects, both symbols, and that for level_of_detail one integer.

Facts take the general form:

```
relation_name ( object1, object2, object3, . . . ).
e.g.
clauses
  colour ( water, blue ).
  colour ( minor_roads, yellow ).
  map_user ( general ).
  map_use ( overview ).
```

The example facts express the English language statements 'the colour of water is blue', 'the colour of minor roads is yellow, 'the map user type is general' and 'the map use is overview'.

Rules link facts together and take the general form

```
relation_name_a ( object, object, . . . ) if
  relation_name_b ( object, object, . . . ) and
  :
  :
  relation_name_z ( object, object, . . . ).
```

Generally the 'if' is replaced by the symbol ':-' and the 'and' by the symbol ','. The relations within a rule may be either facts or further rules, so we can have a hierarchical definition of rules. A rule such as 'if the user of the map is general users and the purpose of use is overview then the level of detail is 4' could be expressed in Prolog as[3]:

```
level_of_detail (Level):-       % rule head
 map_user (general),            % }
 map_use (overview),            % }  rule body
 Level = 4.                     % }
```

Note that objects starting with a capital letter are variables in Prolog. In the example both 'overview' and 'general' are assigned (fixed) values to the symbols in the map_user and map_use facts (i.e. they are not variables). These two facts must be true for the clause to succeed. 'Level' is an 'unbound' variable (i.e. has no value) in the 'head' of the rule (the predicate call), only being assigned a value (bound) in the last line. The value 'bound' to Level in the last line would be returned as its value after the 'body' of the rule has been executed. This bound value for Level could be used as an input variable in the head of subsequent predicates.

In fact, the example does not illustrate efficient use of Prolog and conforms closer to the procedural paradigm. As both map_user and map_use are fixed, this rule could more efficiently be written as:

```
level_of_detail ( 4 ):-
 map_user (general),
 map_use (overview).
```

There would be several alternative versions for such a level_of_detail rule (i.e. multiple clauses) in the program, each returning the appropriate value depending on the truth of the facts for use and user.

In effect the domains and predicates sections of the program are equivalent to the declaration portion of procedural languages and the clauses section is like statements part.

Goals

Prolog operates by trying to solve a goal. Goals can be entered in two ways. There can be a goal section in the program which determines the problem to

[3] Note: anything entered on a line after a % sign is treated as a comment and does not affect execution. Also, predicates can extend over several lines - spaces between terms are not significant.

be solved. The goal looks like any other clause, i.e. it can include both facts and rules. It can simply call one predicate or it can be 'compound' goal calling several predicates in sequence. The goal will only be satisfied if all the predicates called by it can be satisfied. A program can only have one goal, although this is not a major limitation. For example the goal section in MapDesigner takes the form:

```
goal
        initialise,
        mainmenu,
        closedown.
```

Mainmenu is a call to the main menu of the system that repeats until the user selects exit, thus satisfying the predicate and allowing processing to move on to the closedown predicate.

If there is no goal section in the program, the Prolog system opens a dialogue window and requests a goal. To illustrate this, we could rewrite our level_of_detail predicate in a different way:

```
predicates
  level_of_detail ( SYMBOL,          % map user
                    SYMBOL,          % map use
                    INTEGER)         % level of detail
clauses
  level_of_detail ( general, overview, 4 ).
  level_of_detail ( general, analysis, 6 ).
  level_of_detail ( specialist, overview, 6 ).
  level_of_detail ( specialist, analysis, 8 ).
```

Here we have level_of_detail declared with three components, the first two input values and the third an output. We have four different facts for level_of_detail If the goal `level_of_detail ( general, overview, X )` was entered at the goal prompt the system would respond with : `X = 4, no more solutions.`

This simple example can also be used to further illustrate the power of Prolog. We can use the same predicate to carry out different functions. We could for example enter the goal `level_of_detail ( specialist, X, 8 )` and the system would respond with `X = analysis, no more solutions.`

If we asked `level_of detail ( X, Y, 6 )` the system would respond with:

```
X = general, Y = analysis.
```

```
X = specialist, Y = overview.
no more solutions.
```

In this last case, the system has continued to search for all possible alternatives to the goal and reported them.

Although this section has described the goal section, the process is essentially the same throughout the system. The predicates making up the goal are subgoals. Each predicate called by those predicates are subgoals of that and so on. An essential element of Prolog is its ability to follow a line or reasoning until it fails to satisfy a goal (or subgoal). If this occurs, it 'backtracks' to look for alternatives to goals already satisfied. If it finds an alternative it then moves forward again adopting the new values for objects. In order to carry out this backtracking, it is essential that pointers to backtracking points are stored. It is for this reason that Prolog requires relatively large amounts of memory. Everywhere in the program the system encounters a predicate with alternative solutions, e.g. two facts for the same predicate, two clauses for the same predicate, a clause which calls a predicate with more than one solution, etc., a backtracking point is retained.

Without control over the search process, Prolog's 'relentless search for solutions' will continue its blind search until every possible alternative route through the solution tree has been examined. The way this is prevented is to use the 'cut'. The cut (expressed in programs by an !) effectively tells the system that we are satisfied with the solution found to this point and all previous backtracking points for the current predicate can be removed. In our level_of_detail example above, if we entered `level_of_detail (X,Y,6),!.` at the goal prompt we would only get the first answer. Further examples of how the cut operates are given in Chapter 8.

Working with lists

As noted above, lists are an important structure of Prolog. Lists can hold an arbitrary number of objects, so although they appear initially like arrays in other languages, the length does not have to be declared. A list is composed of elements; each element will belong to the same domain. Examples of lists of integers and of strings might be [1,2,3] and ["red","blue","green"]. A list is considered to be composed of two parts: the head, which is the first element; and the tail, which is a list containing all the remaining elements. An empty list is expressed as [ ]. Conceptually a list is a tree structure composed of a head and a tail, the tail being composed of a head and a tail and so on. For example, the list [a,b,c,d] could be written as [a | [b, c, d]], or decomposed even further to [a | [b | [c | [d | [ ]]]]]. The '|' symbol is used to separate the head of the list from the tail.

A list is a 'recursive compound data structure' and as such needs special procedures to process it. The general case is that the list is divided into two portions, the 'head' and the 'tail'. The head element is operated on then the tail, which will itself be split into a head and a tail, and so on. It is generally necessary to have at least two clauses to process a list, one which does the main processing and one which checks to see if the end of the list has been reached.

The classic list operation is to determine if a value is present in a list. This is generally referred to as the member predicate, e.g.

```
member ( name, namelist )
e.g.
member ( blue, [ red, green, blue, yellow ] ).
```

In English the processing of this routine is as follows: 'name' is a member of the list if 'name' is the first element (the head) of the list, or 'name' is a member of the list if 'name' is a member of the tail. The first clause checks the head of the list. If the head does not match 'name', the clause fails and alternative clauses for the member rule are sought. In the second clause, the head (which we now know is not a member) is discarded and the tail processed. The tail then becomes the list used in the member test. In Prolog a simple program might be (PDC, 1992; 183):

```
domains
 colour = symbol
 colourlist = colour*          % define a list of colours
predicates
 member ( colour, colourlist )
clauses
 member(Name,[Name|_]).        % check to see if name is head
 member(Name,[_|Tail]):-       % second clause
       member(Name,Tail).      % recursive call to the member
                               % predicate with the tail
```

If the goal `member(blue,[red green, blue]).` was entered the system would answer 'true'. If the goal `member(yellow, [red, green, blue]).` was entered, it would respond 'false'.

The MapDesigner system makes extensive use of lists. Further examples of list processing are given in Chapter 8.

The Prolog Database

Although all facts and rules can be included in the main program sections, it is convenient to be able to manipulate facts at run time. This is achieved in PDC Prolog by the use of a 'database' section. The database section of the program looks identical to the predicates section and in effect declares the facts which can be stored in a database. These facts can be used in exactly the same way as facts in the main program, but additional facts can be entered (asserted) into the database at run time, or removed (retracted) from the database. Thus, having determined the level of detail using the clauses described above, we may assert its value into the database:

```
predicates
  get_level                                 % get level of detail
                                            % and store in database
  level_of_detail ( SYMBOL,                 % map user
                    SYMBOL,                 % map use
                    INTEGER)                % level of detail
  ask (                                     % asks for information
                    STRING,                 % question
                    SYMBOL)                 % response
database
  l_o_d ( INTEGER)
clauses
  level_of_detail ( general, overview, 4 ).
  level_of_detail ( general, analysis, 6 ).
  level_of_detail ( specialist, overview, 6 ).
  level_of_detail ( specialist, analysis, 8 ).

  get_level:-
        ask ("who is intended map user? ", User ),
        ask ("what purpose is the map for? ", Purpose),
        !,                      % the cut means don't backtrack
                                %     beyond here
        level_of_detail ( User, Purpose, Level )
                                % try and match User and Purpose
                                % with facts for level_of_detail
```

```
            assert ( l_o_d ( Level )).    % add value to database
    goal
     get_level.
```

In practice, the level_of_detail facts could be stored in a database (i.e. the knowledge base) and loaded into the system's memory at run time using the built in predicate 'consult'. Databases modified during program execution can be stored in a file using the 'save' predicate.

PDC Prolog is not limited to a single database. There is always one (and only one) unnamed database associated with a programme, but there may be any number of named databases loaded into memory. Naming a database has no effect on the use of predicates, but by having several named databases (knowledge bases), only the facts relevant to the particular task need be loaded into memory at any given time, hence reducing memory usage for facts and allowing more to be used for backtracking pointers, etc.

Individual facts can be removed from the database using the retract command, although retract will fail if the elements in the predicate call do not match a fact in the database. Whole databases can be removed by using the retractall command (which never fails) and specifying the database to be removed.

Many versions of Prolog allow both facts and rules to be stored in the database, but PDC Prolog only allows facts in the database. This is one of the major limiting factors in developing an expert system where all the domain knowledge is stored in the database (or knowledge base) rather than incorporated into the main program. The limitation can be overcome to some extent in some situations but not entirely. The approach adopted in the system being developed is at least to separate the cartographic rules from the general inference procedures, even if practically the rules have to be part of the program rather than part of the knowledge base.

In summary, PDC Prolog is a 'typed' Prolog compiler which has the advantage of many built in graphics predicates allowing relatively easy development of graphics programs without the need to interface to other languages or systems. A major disadvantage is the limitation that only facts may be asserted into the knowledge base at run time. This makes it harder (but not impossible) for new rules, either inferred by the system or supplied by the user, to be added.

THE INFERENCE MECHANISM

The inference method adopted is a predominantly forward chaining mechanism, i.e. it is a data driven solution, where rules are matched against existing facts to establish new facts. Backward chaining, using hypothesis testing, is used at some points, mainly to establish if a default value is the most appropriate or to make a choice between options. An overall backward chaining approach has been attempted for map design (e.g. Muller & Wang, 1990), but in the author's opinion the lack of adequate evaluation procedures for assessing map design makes this approach less desirable. Merritt (1989; 7) points out that where it is not possible to enumerate all of the possible answers beforehand and have the system select the correct one, a goal driven, or backward chaining, approach will not work. This is because the variability of the inputs and the number of ways they can be combined is almost infinite.

The choice of inference method has some effect on the interface with the user. In a data driven system a number of facts must be initially established for the system to infer new facts from, whereas in a goal driven system facts are only gathered as they are needed. Thus, the initial fact gathering part of the mechanism may appear more like a conventional procedural program, although if correctly programmed only relevant questions will be asked.

A forward chaining system consists of three components: the rule set; a working storage area with the current state; and an inference mechanism which knows how to apply the rules (Merritt, 1989; 74). Rules take the general form:

Rule (no.) IF [condition(s)] THEN [action(s)]

The execution cycle is:

1. Select a rule whose condition(s) match the currently stored state.
2. Execute the action(s) to change the current state.
3. Repeat until no more rules apply.

A disadvantage of forward chaining systems is that it is harder to implement explanation because as each rule is processed it modifies the working storage, thus covering its tracks. 'Why' can be implemented by having an associated explanation for each rule but to answer 'How' some form of trace facility would have to be implemented.

KNOWLEDGE REPRESENTATION

Frames are used as the main knowledge representation method for the map design specification. These are particularly appropriate for problems where we can identify a number of stereotypes, in this case a relatively small number of basic map themes and representation methods. Each of the blanks or 'slots' in the frame must be filled in order. This is achieved by a procedure or procedures being associated with each slot which may, for example, ask the user to answer a question, or refer to other frames (Giarranto & Riley, 1989). The resulting system has a hierarchical structure where the topmost frames represent generalities and the lower ones are customised for more specific instances. Frame based systems have been identified as being particularly appropriate representations of objects in the design process (Lansdown, 1988; 1162).

The structure of frame based systems adds intelligence to the representation by allowing objects to inherit values from other objects. "Furthermore, each of the attributes can have associated with it procedures (called demons) which are executed when the attribute is asked for, or updated " (Merritt, 1989; 9).

In some examples, the inheritance feature of frames is used to reduce the number of slots in an individual frame by using a slot labelled 'a kind of' which indicates that the frame inherits the properties of the frame(s) further up the hierarchy. Slots for these properties are not included in the frame unless the value is different from the default. While this is a more compact representation than fully representing each frame it is less clear. An alternative is to distinguish between frame definitions and instances of frames. Frame definitions specify the slots for a frame and instances provide specific values. for the slots. Frame instances will be updated in working storage and accessed by the rules (Merritt, 1989; 114). For example, **Map** would be a frame definition and **Topographic** would be an instance of Map.

Examples of the MAP frame are given in Figures 7.2 and 7.3. The generic MAP frame (Fig. 7.2) indicates the slots to be filled, the kind of information that will fill them and the procedures that are used to fill them, or that come into operation when the value in the slot is altered. A frame from the second level of the hierarchy is illustrated in Fig. 7.3a. There are three possible frames at this level, for 'basic', 'cultural' and 'physical' maps which exist largely as a starting point for designing maps of topics 'unknown' to the system. At the third level of the hierarchy frames represent individual map topics, and at the fourth and most detailed level frames represent individual map designs, which have all slots filled (Fig. 7.3b). Space does not allow the members of lists or details of the procedures to be shown in these diagrams.

Frame: MAP		
SLOT	**FILLER**	**PROCEDURES**
Title	Optional	Ask_user
Author	*System user*	Ask_user
Date	*System date*	Ask_system
Map_topic	Compulsory	Built_menu
Map_class	Derived from topic	get_class
Notes	Optional	Ask_user
Description		
Map_user	*General*	Menu; if_change_rules
Map_purpose	*Overview*	Menu; if_change_rules
Output_media	*Screen*	Menu; if_change_rules
Level_of_detail	Computed	get_LOD
Layout		
Location	Compulsory	Build_Menu; if_change_rules
Format	*Fill screen*	Menu; when_added_rules; if_change_rules
Scale	*Max possible*	Menu; when_added_rules; if_change_rules
Selection		
Selection_index	Computed	compute_SI; if_change_rules
Base_information	List derived	Selection_rules
Thematic_information	List derived	Selection_rules
Symbol_specification		
Specifications	List derived	Fill_representation_frames

Notes - Filler column - *italics* = default value.

Figure 7.2
The MAP frame

There is also a series of cartographic representation frames (described in Chapter 8). Each of these contains slots for the parameters specifying the symbols in enough detail for them to be drawn. These slots are filled by default values, procedures, or by reference to look-up tables containing colour sequences, point symbols, line patterns, etc.

Frame:	**(a) MAP: Basic**	**(b) MAP: Geological**
SLOT	**FILLER**	**FILLER**
Title	Optional	**Nigeria Geology**
Author	*System user*	**David Forrest**
Date	*System date*	**01/01/95**
Map_topic	**Basic**	**Geological**
Map_class	**Basic**	**Physical**
Notes	Optional	
Description		
Map_user	*General*	**Specialist**
Map_purpose	*Overview*	**Overview**
Output_media	*Screen*	**Screen**
Level_of_detail	Computed	**4**
Layout		
Location	Compulsory	**(2,3,15,15)**
Format	*Fill screen*	**Fill screen**
Scale	*Max possible*	**7500000**
Selection		
Selection_index	Computed	**7**
Base_information	*known list of options*	**[known list]**
Thematic_information	*known list of options*	**[known list]**
Symbol_specification		
Specifications	List derived	**[List of symbols]**

Notes - example columns- *italics* = default value, **bold** = slot filled,
[ ] = list, details omitted.

Figure 7.3 a & b
Examples of MAP frames.

The rules associated with the slots are represented in the system in a number of ways. Due to the difficulties of placing rules in the external knowledge base with PDC Prolog, some are associated with particular modules of the inference mechanism. Those procedures which are basic to the system and would apply universally to any cartographic design system, such as those for calculating and checking scale and format, are integrated with the inference mechanism. Others more specific to this system are separated to allow for easy modification.

As the size of the knowledge base grows, performance becomes problematic. Various indexing systems can be used to speed up the process required to find the next rule to solve. The rules in forward chaining systems generally need to be indexed by more complex methods than required for backward chaining systems. By using the MAP frame definition as the controlling structure for inference and filling the slots in a specific order to create instance frames, complex rule indexing methods may be avoided initially. They may be required if full flexibility is to be achieved at the Modify stage of design. The problem is also reduced in MapDesigner by declaring a number of different rule and fact structures for specific aspects of the problem, rather than trying to use generic rules and facts to cover all situations. This is essential as the problems to be solved for different slots are unique to that stage in the process. The whole nature of the problem, and hence the solution, is quite different to the 'classical' classification type of expert system typically described. A possible future development would be to incorporate a version of the Rete match algorithm as described by Merritt (1989; 138) used in other forward chaining systems to optimise performance.

Although the order of processing the slots is fixed in the inference engine, any slot may in fact have its value pre-assigned, i.e. the system can load partially complete frames. Before calling the procedures associated with a slot, a check is made to see if it is already filled.

THE USER INTERFACE

The user interface to the system is menu based. The user is never confronted with a blank screen and left wondering what to do next. Two basic screen arrangements are used, one during text based queries and the other when graphics are being displayed. This is due to the simpler handling of textual information with PDC Prolog in text mode rather than graphics mode. The user interface toolbox that is supplementary to the main Prolog system provides a wide range of facilities such as different types of menus, but only for text mode operation.

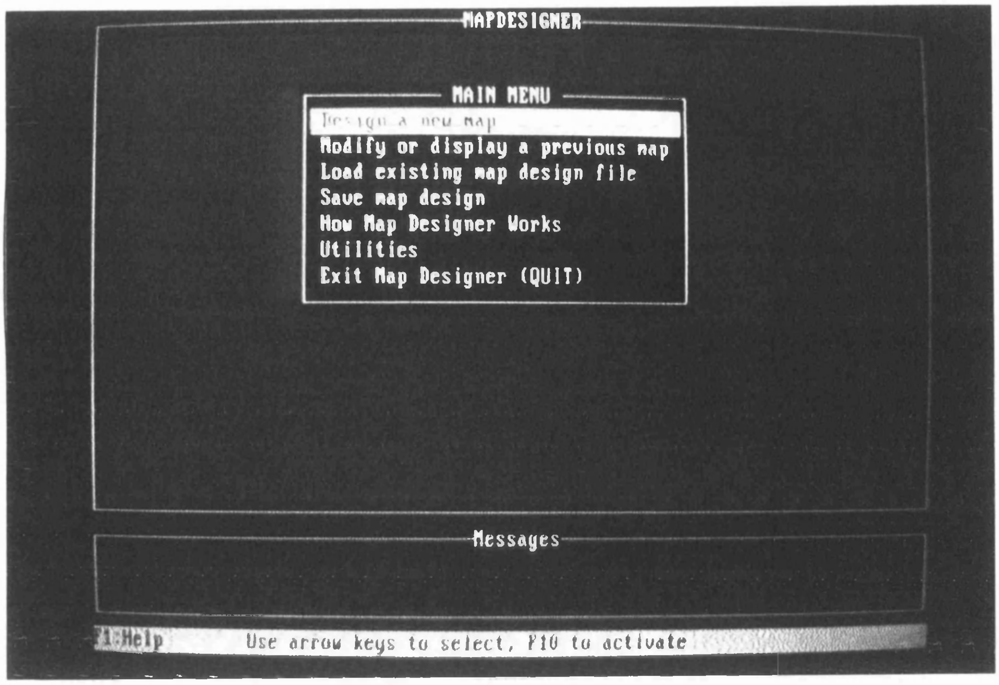
Responses to menus are entered via the keyboard. Mouse support is available in the toolbox, but has not been implemented for two reasons. First the systems used in the initial development of MapDesigner did not have a mouse and second, the types of response required can be quite satisfactorily achieved using the keyboard. It would be quite possible to add mouse support as the predicates in the user interface toolbox are designed to allow this.

The screen layout is similar throughout the different modules of the program and is made up of three areas in text mode (see Figure 7.4). and four in graphics mode. At all times the bottom line is a status line. This indicates the action required by the user, such as 'press space bar to continue', and the other keys which are active, such as F1 for help, etc. Immediately above the status line is a message window. This is used to inform the user about current actions by the system, or issue warnings and error messages. Simple messages such as 'loading knowledge base - please wait' may appear and disappear without the user being aware of them - no particular attention is drawn to them and they largely are issued to inform the user that something is happening. If an error occurs (e.g. the user has entered an illegal value) the message is accompanied by a beep and the system pauses until the space bar is pressed (indicated in the status line) to acknowledge the message.

The largest part of the screen in text mode is used for the main dialogue with the user. The current module name is indicated by a title at the top of the box surrounding the dialogue area. These modules are colour coded (e.g. Description has a green frame and uses green highlight boxes), although most users will probably be unaware of this. The dialogue area has a blue background with normally yellow text. Menus appear within this area and have a black background with white text.

Most of the interface simply requires the user to move the cursor to the appropriate menu item using the cursor keys and then press the F10 key to activate. Occasionally the enter key has to be used and the space bar after any pauses in execution to allow the user to read the screen. The F1 key is used to call the help facility, the F2 key to ask 'Why' a question is being asked, the F3 key for 'How' decision has been reached and the Esc key to terminate the current activity. Wherever possible default values are given to questions (both with and without menus) so the user only needs to press the F10 or enter key to accept that response.

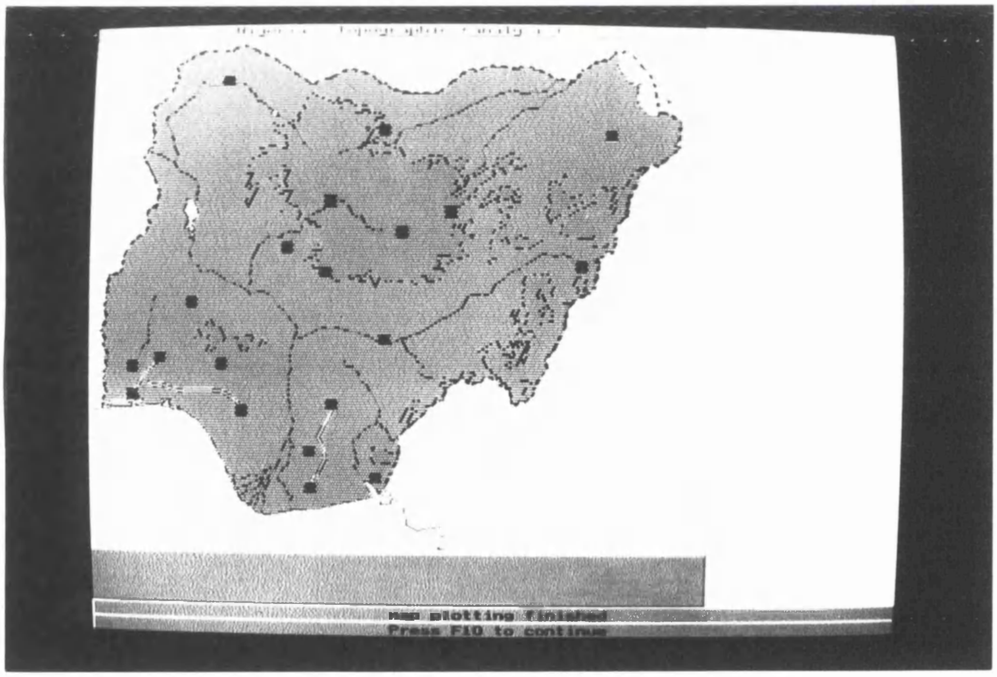
The majority of the menus used simply list the options and require the appropriate choice to be made. Some menus allow for multiple selections and, for the selection of map topic, a tree structured menu is used rather than a simple list. The construction of this tree menu is discussed in Chapter 8. The interactive screen handler of the toolbox is also used where more varied input is required, such as in entering the limits of the area to be mapped. This allows the user to enter (or edit) values in a number of boxes in the screen. When the F10 key is pressed the values in each of these boxes is passed to the program. The use of this technique is particularly useful in editing existing parameter values.



```
MAPDESIGNER

MAIN MENU
Design a new map
Modify or display a previous map
Load existing map design file
Save map design
How Map Designer Works
Utilities
Exit Map Designer (QUIT)

Messages

F1:Help    Use arrow keys to select, F10 to activate
```

Figure 7.4

An example of the standard text screen layout.



```
Map plotting finished
Press F10 to continue
```

Figure 7.5

Example of graphics screen layout.

In graphics mode, the status and message windows are similar, but the main part of the screen is divided into a map window and a legend window (Figure 7.5), although currently no legend is plotted. This rather simplistic approach to layout is satisfactory to illustrate the operation of the system, but clearly better use could be made of the available space depending upon the shape of the map, or indeed the outline of the area of interest. Such matters of overall composition of the image are obvious areas for future development.

Help, How and Why

Some form of help system is an important part of any fully developed user interface. Although no help system is currently implemented in the program, the necessary links for a comprehensive context sensitive help system are built into the user interface toolbox predicates, which is one of the reasons for adopting these for all input from the user. All that is required to implement a help system is to provide a file of the necessary help information and to set the correct help context before each opportunity for the user to enter information. If the user then presses the F1 key the appropriate help message would appear. On clearing the help message, execution would proceed as normal.

Explanation facilities are much harder to implement, particularly, as noted above, for forward chaining systems. In asking 'Why' the user is seeking information about a question - why is it being asked; in asking 'How' he is seeking information about an answer - how has that result been determined. The difficulty in implementation probably accounts for the almost total lack of mention of explanation facilities being implemented in the literature on cartographic design expert systems.

It is easier to include a system that appears to answer 'why' than 'how'. Many 'why' questions can be answered in a similar way to the context sensitive help system described above as one can predict the situations where these questions are likely to be asked and incorporate standard answers. Such an explanation is included in the system developed by Wang (1992), but this is not a true explanation system as it does not track the logic of the program as it is executed. Although one could try and predict where 'how' would be asked and provide stock answers - basically textbook definitions of standard situations - this is not a satisfactory solution. Also, all discussions in the literature refer to textual explanations of questions or solutions. A comprehensive explanation facility for a cartographic design expert system would also need to incorporate graphical explanations. This possibility clearly demands further investigation.

For certain parts of the system it would not be too difficult to track events and provide satisfactory textual explanations, particularly the description and layout modules, but where questions would be more significant is in the symbolisation module. Here the extensive use of recursion and backtracking and the number of symbolisation possibilities resulting in the combinatorial expansion of the search tree make it much harder to implement satisfactory explanation facilities. As a result the system does not incorporate an explanation system, although like the help facility links exist in the program to allow one to be implemented at a later date.

DATABASE & METADATA

The database

Although the system is capable of producing a specification for an abstract map, in order to produce an actual map it is first necessary to have some information to map. In the longer term it is desirable that the system be capable of interfacing with a number of databases, but it is expedient to limit the interface to a customised database in the first instance. Experience has shown that PDC Prolog is very particular about the format of files to be read and is prone to problems in reading 'foreign' files. The preferred situation is for PDC Prolog to read files it has previously written. It is possible to simply read in lines from text files and carry out type checking and conversion, but this is a diversion from the main task of the system, so the most straight forward solution was adopted in order to speed development.

Database files read by the system are all of one of four formats: node files which contain point information; chain files which contain line data; polygon files which contain the boundaries of areas; and data files which contain attribute information about points or areas (Currently all lines used by the system have either a feature code or a single value and therefore do not need to be associated with a datafile) . The format of each of these files is given in Appendix E. In many current GIS, chain and polygon files are equivalent and the polygon boundaries are built from topologically encoded chains, but polygon construction has not been incorporated into the system as developed, although the specification of chain files is such that they could be used to build polygons if required.

In order to match the requirements of the Borland Graphics Interface (BGI) plotting routines in PDC Prolog all chains and polygons are read as 'terms', i.e. the whole string of co-ordinates is read as a single entity (a list). In practice, this places a limit on the maximum number of co-ordinates in any chain or polygon (although no details of limits for list membership are given in the manuals). The main difficulty is getting the data into a file written by PDC Prolog. In building the database it is necessary to reconfigure compiler memory allocation to enter large data sets, but it is not possible to run the map design system with this configuration. Once read in and stored as Prolog terms there seems to be no difficulty in accessing, manipulating and plotting the information.

Metadata

There are some differences in the use of the term metadata in current literature on GIS. In some cases it is used to refer to rather general, global characteristics of databases or data sets, containing information about sources, availability, dataset quality, etc., to enable potential users to evaluate their utility before purchasing or otherwise acquiring the data. The alternative use refers to a more detailed description of individual elements of the dataset and is perhaps more an extension to the data dictionary incorporated into most database management systems. Although there is obviously overlap between these two levels of knowledge, the differences are significant. Here the term is used to refer to the more specific information describing in some detail every element in the database.

As mentioned previously, a long term goal is to develop an expert system capable of interactively building this metadata or knowledge base about the data, shown in Figure 7.1 as the Geographical knowledge acquisition sub-system. Currently the knowledge has been determined manually based upon analysis of the data using the guidelines set out in Chapter 5 about phenomena and information. As the system developed, the importance of this metadata became increasingly obvious, although it is rarely if ever discussed in developing design expert systems, where the emphasis is usually on representational issues. Having determined the metadata characteristics these are entered into a series of facts in a knowledge base called 'Meta'. The expert system to be developed to acquire this knowledge would quite likely use a frame based model as again there are stereotypical situations, and the storage structure used by 'Meta' would allow this approach.

Four different types of metadata are stored in the metadata knowledge base. These are the main information about classes of data, details of individual co-ordinate files, details about data (attribute) files and lookup tables to translate between names for features used in the system's knowledge base and their names in the database. These are referred to as meta_data, coord_file, data_file and look_up respectively. Their structure and possible contents are illustrated by Figures 7.6, 7.7, 7.8 & 7.9.

SLOT	Possible values ( ) = comment
feature category	(symbolic name for feature class used in MapDesigner expert system)
data capture date	
comment	
phenomenon	discrete (point at map scale) linear specific areas continuous surface
spatial data	points lines boundaries (polygons not explicit) polygons cells
attribute level	identical feature coded hierarchical_feature_coded ordinal interval ratio_absolute ratio_density ratio_derived external (not defined here)
nature of phenom	tangible conceptual
symbolic name of co-ordinate file	(reference to 'coord_file' not actual name)
lookup file name	(actual file name)
symbolic name of data file	(reference to 'data_file')
symbolic name in data file	(the name of the attribute or column)

Figure 7.6

Meta_data slots and possible values.

SLOT	Possible values () = comment
symbolic name of file	(provides link with meta-data)
file type	point line boundary polygon cell
digitising date	
coord limits of data	
digitising scale	
source	
projection	sphere (i.e. coords in Lat & Long) plane projection_name
units	degrees miles kilometres
scale factor for coords	(multiplier of file co-ordinates to get units)
filename	(actual name of file stored on disk)
comments	

Figure 7.7

'coord_file' metadata and possible values.

SLOT	Possible values () = comment
symbolic name of file	(provides link with meta-data)
file type	data (only attribute values) coord (both co-ordinates and attributes)
data source	
variable names list	(list in order of columns in file)
filename	(actual name of file stored on disk)
format	(generally Prolog terms, but may be specified)
comments	

Figure 7.8

'data_file' metadata and possible values

SLOT	Possible values () = comment
feature class	(class name in MapDesigner)
feature kbase name	(name in MapDesigner knowledge base)
feature dbase name	(name in database)

Figure 7.9

'look_up' metadata and possible values

It should be noted that as currently implemented by MapDesigner, the datafile may consist only of data or may include co-ordinates as well as data. The data format used for point data allows one feature code and one numerical value for each point to be included in the file. Polygon and chain records both include a field for a feature code (See Appendix E). This means that the symbolic names in 'meta_data' for 'coord_file' and 'data_file' may be the same. The system checks for this and only opens the appropriate file once. If no 'data_file' is specified in 'meta_data' the system assumes that the feature code in the co-ordinate file will be used.

The fourth set of metadata, the lookup table of feature names allows greater flexibility in linking existing data with the system. MapDesigner makes extensive use of descriptive names for features, e.g. major_river, state_boundary, whereas in digitising different feature codes may have been used (for example MapData uses numerical codes). It would not be practical to convert all existing datasets to the MapDesigner names so the 'look_up' metadata provides the linkage. Where the database name is the same as the MapDesigner name no look_up file need be specified in 'meta-data'.

This chapter has outlined the basic structure of the system and its major components. It has also introduced some of the basic principles of programming in Prolog. The next chapter describes how these principles have been applied to the particular problem and gives details of the solutions to the tasks involved.

CHAPTER EIGHT

Description of the MapDesigner System

> While it is possible to download information [to a computer] ... this would not retain the essential experience. There is an ineffable quality to [human] memory which cannot survive this procedure.[1]

THE BASIC STRUCTURE

As was shown in Figure 2.1 the basic components of a cartographic expert system are the knowledge base, the inference engine, the user interface and a database of spatial and attribute data. MapDesigner conforms to this basic pattern, although due to limitations of the Prolog compiler the separation of the knowledge base from the main program or inference engine is not complete. Where elements that would ideally be part of the knowledge base are incorporated into the program modules they are generally kept separate from the inference mechanisms and any parts that can be accessed from the knowledge base are. This results in an inference engine that is more specific to this particular problem than would be the ideal case, but there is still sufficient flexibility in the system to extend the range of scales or topics which could be accommodated.

The system is of considerable size and is broken down into modules. This is essential due to the memory limitation of PDC Prolog, but also means that the program is split into logical units and a single unit can be developed independently of others. The main disadvantage of modular programming is that it is not possible to retain backtracking points across module boundaries. This means, for example, that when choosing representation methods, if the range of features selected cannot all be assigned a representation method the system should backtrack to the selection process and reduce the number of features selected. Currently the system would fail in this situation and the user would need to interactively reduce the number of features selected. In practice, testing of the system has never resulted in such a failure and it is thought to be unlikely that this will occur given the way the system prevents conflict at the selection stage. Problems are most likely to arise when the user requires several thematic topics to be included in the one map. Two such topics is probably the practical limit of the current system (e.g. urban and rural population; temperature and precipitation). Beyond this the map is likely to be rather complex. There is body of opinion against including too many topics in one

[1] Lt. Cmdr. Data [an android], Star Trek: The Next Generation, 'The Measure of Man.'

map as they become too complex to communicate their message. It is probably better to carry out some analysis of the information and map the results of that analysis, or to produce a number of simpler maps.

There are more modules than the main sections indicated in the discussion on the map design process in Chapter 5. This is due to memory limitations and primarily affects the symbolisation stage which is further broken down into sub modules. There are also additional modules such as a utilities module and there are links built into the main system to allow further expansion by including modules for data verification, editing the knowledge base, etc.

In parallel to the modular structure of the inference engine, the knowledge base is also split into modules. While this is not the most desirable scenario, it is again due primarily to memory constraints, but also simplifies the loading and removal of facts relevant to the current program module. The sectioning of the knowledge base does not affect operation in any way, other than using separate predicate names for the different modules where perhaps one predicate could be used more extensively.

The intention here is not to describe each and every minor step in detail, but to follow through the process of designing a map and outline the main function(s) of each module and show how the various tasks are solved. This is followed by a more detailed discussion of some specific cases and examples of system output.

In the program extracts included here the following should be noted:

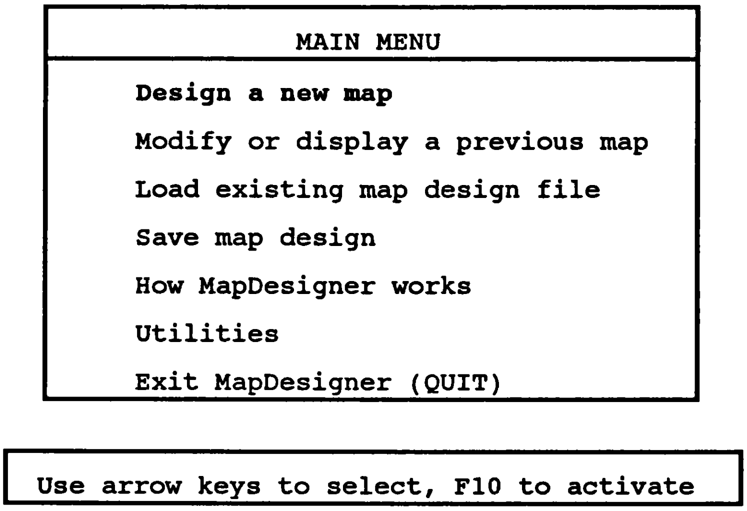
1. the extracts are not necessarily complete operational examples, but an indication of the structure of predicates and clauses.
2. many predicate names in the actual programs have an additional prefix to help identify their context, e.g. all knowledge base predicates are prefixed by 'k', frame slots by 'f', temporary working values by 'w', etc. These prefixes are generally omitted here for ease of reading unless needed to distinguish between predicates.
3. Prolog conventions such as the use of [ ] to identify lists, :- to represent 'if', etc. are used.
4. predicates called within a given clause are indented. Clauses end with a '.'. Prolog statements can extend over several lines, all blank spaces (except between quotes) being of no significance, or several predicates can be on the same line.

5. the underscore character _ is used to join characters to form a single identifier in Prolog and is widely used to increase readability. It can also be used on its own to represent the 'anonymous' variable, i.e. it will 'match' any value.
6. anything on a line to the right of a '%' is a comment and is ignored during compilation.
7. the menus shown here represent the main element of the screen. However generally there will be other information shown as to how to use the menu, etc. Items in bold in these menu (reversed out text in system) indicate defaults.

Full listings of the programs, knowledge bases, etc., are given in the appendices, as are screen images of some of example menus.

MODULE 'MAIN'

This is largely a housekeeping module. It issues messages welcoming the user, reads a set-up file (setup.kba) and checks to see that the knowledge bases are valid. The main menu is presented to the user who selects the appropriate task (Figure 8.1). The main menu operates in a similar fashion to most of the menus presented to the user with the relevant instructions on how to use the menu being given in the status line at the bottom of the screen. The default choice is highlighted and can be selected by pressing the F10 (or Enter) key. Other choices are made by using the arrow keys to highlight the appropriate choice then pressing F10. It should be noted that at no stage is the user confronted with a blank screen: there is always some instruction as to how to proceed.



MAIN MENU

Design a new map

Modify or display a previous map

Load existing map design file

Save map design

How MapDesigner works

Utilities

Exit MapDesigner (QUIT)

Use arrow keys to select, F10 to activate

Figure 8.1
The Main menu

The main controlling predicate 'design_map' is part of this module. This predicate calls each of the modules required to produce the map in turn. Once a map has been designed the user is returned to the main menu. When exit is selected all open knowledge bases are closed, the most recent map design specification is written to a default file (frame.frm) and execution stops.

Design_map

This is the main predicate which builds up the map specification frame and plots the map. When entering each of the main predicates called by design_map, a check is made to see if the appropriate slot is already filled. If so, the system moves on to the next slot. This allows partially specified maps to be loaded by the system and completed. Thus, if the user knows he wants a particular feature included and/or symbolised in a certain way this can be pre-set. This does not alter the knowledge base, but does allow, for example, a standard base map to be specified for a series of maps. It also allows one mechanism for editing previous maps whereby the values to be changed are simply deleted from the frame. This chapter concentrates on the modules called by this predicate.

Other functions

As noted, many of the functions of the Main module are of a housekeeping nature. Several of the functions called from the main menu are still to be implemented, such as loading existing map design frames and explicitly saving them. Saving the most recent frame occurs automatically to a default file called 'frame.frm', although this is only intended as a mechanism for checking the operation of the program, and clearly any released version would have to implement these functions. 'How MapDesigner Works' is intended to provide a brief tutorial on the system.

Utilities

Selecting 'Utilities' from the main menu brings up a further menu which allows one to: edit the set up file; edit the knowledge base; load different knowledge bases; or carry out operating system commands. Apart from the last on this list, these functions have not been fully implemented in the current version of the system.

MODULE 'DESCRIPTION'

This module largely gathers preliminary information about the user, the map topic, etc., and as such uses a forward chaining (data gathering) mechanism. It provides values for the slots map_date, map_author, map_title, map_topic, map_purpose, map_user, output_media and level_of_detail. In any run of the system the maps produced are automatically allocated a sequence number. This ensures that all maps within a given run of the program are unique. Previous map frames from the run can be selected for editing in the Main module (as well as loading frames from previous sections).

Although this module provides values that fill more slots than any of the others, it is the least complex. Map_date is automatically filled by the system. Map_title and Map_author are both filled by the user entering the desired name when prompted, null values being acceptable (i.e. no title or author designated). Map_purpose, map_user and output_media slots are all filled by selecting from simple menus, whereas level_of_detail is obtained from the knowledge base.

The module ends by displaying a summary of the slots filled by this module.

Map_User & Map_Purpose

These two slots are filled by values returned from pre-defined menus. The map user is by default the system user, with alternatives of general users or specialist users being presented. Maps for general users will have less detail; those for specialists or the map author (system user) will have more. The default for purpose is 'overview' with the only alternative being 'analysis'. Maps for analysis will contain more detail; maps for general overview will be simpler. Thus the least detailed maps will be overview maps for general users and the most detailed will be maps for analysis by specialist users.

Map_Topic

The map_topic[2] slot is a critical slot as it is one of the primary controls over map content. The more closely the user can choose the main map topic to his desired product, the more the system can help, or complete the process automatically. The menu presented to the user is in the form of a tree rather than a

[2] This slot was previously referred to as map_type, and this name is used in the programme. At a late stage it was decided that map_type could lead to confusion and therefore either map_theme or map_topic was preferable.

simple list of topics. The further to the right a topic is in the tree the more detailed it is defined.

This menu is not pre-defined; the tree is built automatically for each run from information in the knowledge base (see 'kbase.kba' in Appendix B), and consequently it is a simple matter to add new map topics to the system without requiring recompilation of the inference engine.

The fact kmap_content essentially represents a partially filled Map frame or 'child' of Map for each of the known map topics. It takes the form:

```
kmap_content(map_topic, map_class, [list of base_info scores],
  [theme_info_list], base_info_list)
```

Map_topic is the primary subject of the map, e.g., topographic, population, precipitation. Because each topic belongs to a higher level class (map_class), a hierarchy of map classes is defined and a tree structured menu built as shown in Figure 8.2. The list of scores for base information (currently 18 features, but modifiable in the knowledge base without recompilation) indicate the probability of including each potential base map feature in a map of that topic. A feature with a score of 10 means that the feature should always be included in maps of the selected topic and a score of 0 means never include it. The names of these base information features are stored in a list with the name given by 'base_info_list', the names being listed in the same order as the scores. Therefore it is possible to have different knowledge bases using different sets of base information. Further detail of the map_content predicate describing how selection is carried out is given below.

'Theme_info_list' is a list of the sets of topic information that could be expected to be included in a map of that topic. If the topic is well defined (to the right of the tree) e.g. 'urban population', a single theme data set (or none) will be included. For less well specified map topics, e.g. 'population', a number of themes or variables will be included. In this case further questions will be asked later to determine which of the possible themes are to be included. (Potentially, new variables could be added or computed from the database, but this is currently not implemented).

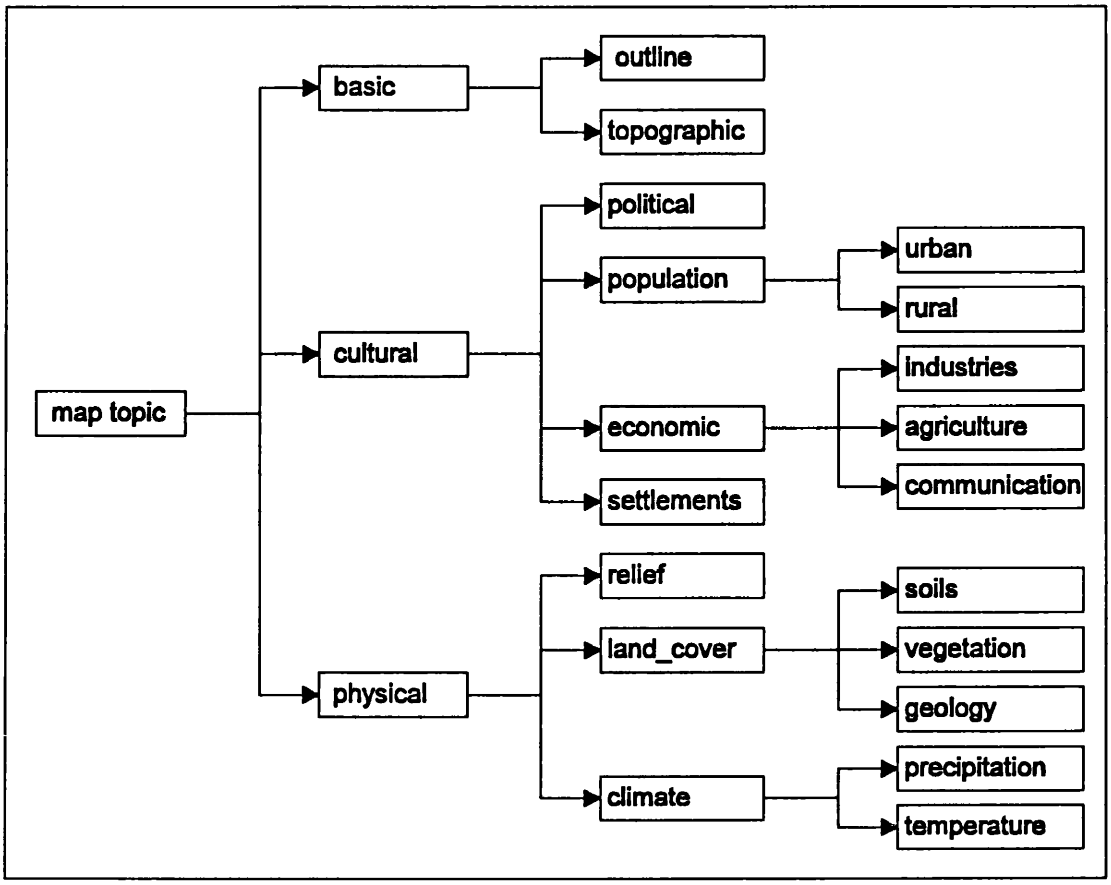


Figure 8.2
Map Topic menu

Output_media

This important slot is filled by a value returned from a pre-defined menu. Various options are presented, including screen, monochrome print, colour print, slide, etc., but currently only screen can be selected. Although there are many parallels in designing maps for different media, there are differences in resolution, colour etc., to be considered and certain parts of the knowledge base are specific to the medium, particularly those parts connected with the specification of symbols and the display module. In a fully developed system different knowledge bases would be loaded for different output media, but the prototype is limited to screen maps.

Level_of_detail

This slot is derived from the values of map_user, map_purpose and output_media. Currently there is a series of facts in the knowledge base covering the possible permutations of these values, each with a value attached:

```
level_of_detail( map_user, map_purpose, output_media, l_o_d).
e.g.
```

```
level_of_detail( "author", "overview", "screen", 4).
level_of_detail( "general", "overview", "screen", 2).
level_of_detail( "specialist", "analysis", "screen", 8).
```

With three values known (map_user, map_purpose and output_media), the unification function of Prolog searches for the appropriate fact that matches the bound input variables and binds a value (l_o_d) to the unknown variable:

```
get_levelofdetail :- flevel_of_detail(_),!.
                               % value already in frame
                               % don't do any more
get_levelofdetail :-           % main clause
 map_user(User),               % get user from frame
 map_purpose(Purpose),         % get purpose from frame
 output_media(Media),          % get media from frame
 !,                            % cut - having got these values
                               % don't look for alternatives
 klevel_of_detail(User, Purpose, Media, LOD),
                         % look in kbase for matching fact
                         % this will give a value to LOD
                         % which was unbound
 assert(flevel_of_detail(LOD)). % put value into frame slot
```

It would be possible to contrive a method of computing the level of detail, but a simple solution described above was adopted. The level of detail can range from 1 to 10, although for screen maps the maximum value returned is 8 due to the limited resolution of the screen display. Output media such as slides or overhead foils would normally have a more restricted range, whereas maps designed for print would use the full range. Monochrome output would have less detail than colour.

Although this value is derived automatically, the user has the opportunity to modify it in the 'show description' screen.

MODULE 'LAYOUT'

This module is responsible for filling slots determining the location, the format and the scale. No attempt is made to produce anything other than a rectangular map area. Overall map composition is not considered, although space is reserved on the screen for titles, legend, etc., in standard positions. Again, this module ends by displaying a summary of the information gathered.

Lat_long

As the system is directed to small scale maps, latitude and longitude are seen as the primary means of defining the area of interest, although the user may opt to enter projection co-ordinates of the area (see below). Ideally the user would be able to have an outline map displayed on the screen and pick the desired area with the mouse, but this has not been implemented. An alternative is to let the user choose the area of interest by name from those entered into a gazetteer and this possibility is offered as the default choice. The gazetteer is organised in a hierarchical fashion which first shows continents, then countries then states or regions. The menu allows multiple zones to be selected simultaneously (rather than having to repeat the menu several times). Between levels in the hierarchy of places a menu requests whether all sub-zones or only a selection are required. Having selected the required zones, the system computes the maximum and minimum values for latitude and longitude from the individual zone limits.

Facts in the gazetteer take the form:

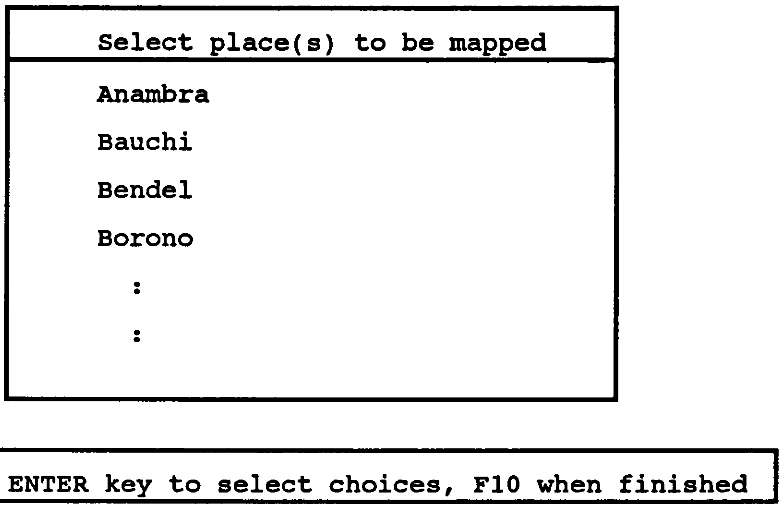
```
extent ( Zone_name, zone_class, min_long, min_lat, max_long,
                    max_lat).
e.g.
  extent ( "Africa", "World", x1, y1, x2, y2).
  extent ( "Nigeria", "Africa", xm, ym, xn, yn).
  extent ( "Anambra", "Nigeria", ...        ).
  extent ( "Bauchi", "Nigeria", ...         ).
```

The list of places for the menu at the appropriate level of the hierarchy is assembled using the Prolog predicate 'findall', e.g.

```
findall ( Place, extent ( Place, "Africa", _,_,_,_), Places)
```

This assembles the list 'Places' which contains all the values for 'Place' which have "Africa" as the second argument in the predicate 'extent'. (In practice, Africa could have been passed from the higher level menu call listing all the continents.) "Places" would then be passed to the menu. This again illustrates the principle that wherever possible information is removed from the controlling program, and that menus, etc., are only constructed when they are needed and that what is in the menu depends on the knowledge base attached, i.e. it is independent of the inference engine.

The location selection menu uses the Prolog Toolbox 'Menumult' which allows multiple values to be returned from the menu, rather than just one (Figure 8.3). If the user fails to select a zone from the list, the system automatically repeats the menu with the warning message that somewhere must be selected (i.e the system doesn't just crash). The returned list from the menu is the position of the items chosen from the input (displayed) list, not the actual values (names). This index list is then processed to extract the names of the zones required from the 'Places' list. This resulting list of the desired zones is then scanned, the gazetteer searched for the maximum and minimum extent of each and the overall maximum and minimum latitude and longitude computed.



```
Select place(s) to be mapped
Anambra
Bauchi
Bendel
Borono
   :
   :

Use ENTER key to select choices, F10 when finished
```

Figure 8.3
The select places menu

Further additions to these options would be to allow the user to specify certain places (i.e. 'point' features) to be included or to specify a radius around a point. This would be simple to add to the system at a later date.

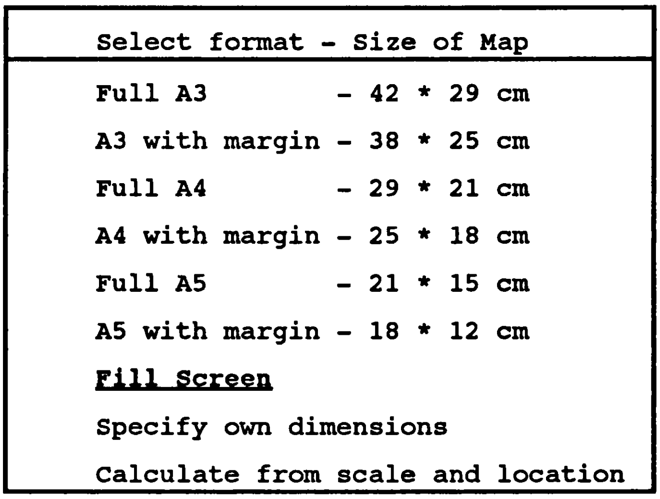
Limits

The limits are the minimum and maximum co-ordinates in the selected projection. If latitude and longitude values are entered then the limits are calculated by the system. The module has the ability to use any number of projections, but currently only one is included, reflecting the projection of the test data set. It is beyond the scope of the present exercise to develop a map projection selection module although in a fully developed system one of the map projection expert systems described in the literature could be incorporated.

It is a requirement that map limits are specified, and this must be done before the scale or format is determined. It is not possible to produce map specifications without defined limits.

Format

Once the limits have been determined, if either the format or the scale are known the other can be calculated. Either format or scale could be determined first, but it is likely that there will be constraints on format, due to screen size, plotter size, report format, etc., therefore, if neither is pre-defined, the format menu is shown first. To assist users in specifying the format, a list of common formats is given (Figure 8.4), although the user is allowed to specify any dimensions required or to opt to specify scale first. When the output is to the screen, the default setting is 'fill screen'. For plotter output the default would probably change to 'A4 with margin'. The system automatically determines if portrait or landscape format is more suitable.



```
Select format - Size of Map

Full A3          - 42 * 29 cm
A3 with margin   - 38 * 25 cm
Full A4          - 29 * 21 cm
A4 with margin   - 25 * 18 cm
Full A5          - 21 * 15 cm
A5 with margin   - 18 * 12 cm
Fill Screen
Specify own dimensions
Calculate from scale and location
```

Figure 8.4
Format menu

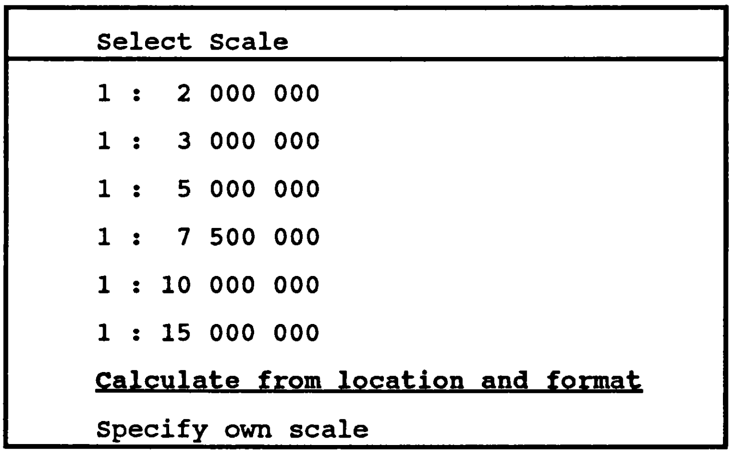
Clearly an 'intelligent' system must be able to check if the selected format fits within the allowable format of the output device. Many systems do include this facility for printers and plotters, but most systems, both mapping and other graphic packages, handle the size and scale of screen images very poorly. It appears to have escaped the attention of software developers that the image area on a 17" monitor is bigger than that on a 14" monitor. Generally the only set up parameter that can be accessed by the user is the screen resolution, but a virtually univeral assumption is made that a 14" screen is in use. For example, in all Microsoft

Windows based software the author has used, selecting an image view of 100% (i.e. life size) gives a larger than full size view on a 17" monitor. If it is possible to inform Windows of the monitor size, this facility is not easy to find. The monitor size, therefore, should be specified in the program set up information, in the same way as the size of a printer or plotter would be, in order that a check can be made on the maximum size of map that can be shown at the selected scale.

Scale

If the format has been specified first the scale is computed. If this is outwith the scale range allowed by the current system the user is prompted to reselect a suitable format. The allowable scale range is specified in the knowledge base and reflects the range of scales for which the knowledge base has been developed. It is not a limitation of the inference engine. Although the exact scale is calculated initially, the value entered into the slot is the nearest smaller scale from a list of sensible rounded scales stored in the knowledge base.

If the format has not been specified, a menu is displayed listing a selection of rounded scales within the range allowed by the system (Figure 8.5). The user may select one of these or enter any specific scale. The format is then computed. If this exceeds the known dimensions of the output device the maximum possible scale for the selected limits is computed and reported to the user.



```
Select Scale
1 :  2 000 000
1 :  3 000 000
1 :  5 000 000
1 :  7 500 000
1 : 10 000 000
1 : 15 000 000
Calculate from location and format
Specify own scale
```

Figure 8.5
Scale menu

MODULE 'SELECTION'

As the name implies, this module is concerned with selecting the appropriate datasets to be plotted from those available in the database, based upon the information gathered above together with facts from the knowledge base about what should be included in the desired kind of map. The selection process operates at the level of classes, and also at the sub-classes level when such hierarchical information is available, i.e. it will choose the appropriate level of the hierarchy to include but will exclude less important levels. It does not currently selectively omit (i.e. eliminate) members within a sub-class.

Selection Index

In order to include the appropriate level of base information, the first stage is to compute a selection index. Values for scale and level_of_detail are retrieved from the current frame and the maximum scale is retrieved from the knowledge base. The formula used to compute the selection index is:

Index = 11 - (trunc {(max scale number / selected scale number) * 10}
+ {(level_of_detail - 5) / 2}).

This formula was developed empirically and has produced satisfactory results over the scale ranges involved. Alternatives may be required for different scale ranges and this is a topic that merits further investigation.

The predicate returns a value in the range 1 to 10. The larger the scale, or the higher the level of detail required, the lower the index value. The index value is then checked against the list of scores for base information in the 'map_content' predicate (see Map_Topic above) and those feature classes with scores equal to or above the index are selected. This assumes that the scores were determined with a knowledge of this criterion, although the critical element is the maximum scale for which the knowledge base is designed.

Base_info_list

The user may opt for either automatic or manual selection of base information. Currently only the default, auto selection, of base information is implemented. It would be a trivial matter to let the user select their own base information and this should be possible in an operational system. However, this is a situation where the system can easily remove a task from the user, and one which he will rarely be greatly concerned about, thus leaving him to concentrate on more difficult tasks. Even with manual selection it would be useful for the system to highlight the set of preferred choices.

The classes of base information stored in the database are given by the fact:

```
klist ("base_info_types",
["Coastline", "Major Rivers", "Large Rivers", "Other Rivers",
"International Boundaries", "State Boundaries", "Tertiary
Boundaries",
"Capitals", "Main Towns", "Minor Towns", "Urban Areas",
"Main Highways", "Highways", "Other Road", "Railways",
"Main Relief", "Minor Relief"]).
```

These are given in the same order as the numerical values given in the list 'base_info_scores' in the 'map_content' fact. The names on the list are used internally within MapDesigner to refer to these features. The feature codes used for these features in the database may be different, look-up tables being used to relate the two (see Display Module below).

The assumption is made that these are the most likely classes of base information to be available for inclusion in small scale maps of the types under consideration and will be in the database. It is possible for any of these features to be absent from the actual database. The system will not fail if it is unable to find data for them. It will however assume that they are present during the symbolisation phase and therefore the map specification (i.e. the frame) will include them. It is quite possible to link the system to different geographic databases with different contents, although ideally the map_content facts should closely match the database contents.

The list of scores in 'map_content' is scanned and features selected which have a score greater than or equal to the selection index. Both the feature name and its score are stored in the frame. It is important that the names list and the scores are in the same order as all future references to base information are taken from the names stored in the frame.

Although automatic selection should not result in inconsistencies, further checks are made on the base features selected to ensure that where a feature depends on another one being present that this is in fact present. This uses a knowledge base predicate that states if 'a' exists then 'b' is required:

```
required ( context, a, b, probability)
e.g.
 required ( "baseinfo", "Large Rivers", "Major Rivers", 5).
        % if large rivers are required then so are major rivers.
```

The probability can range from +5 (absolute certainty) to -5 (no possibility or exclusion). In this case values below 3 are of little relevance and currently it is assumed that in this situation only facts for positive probabilities are entered into the knowledge base.

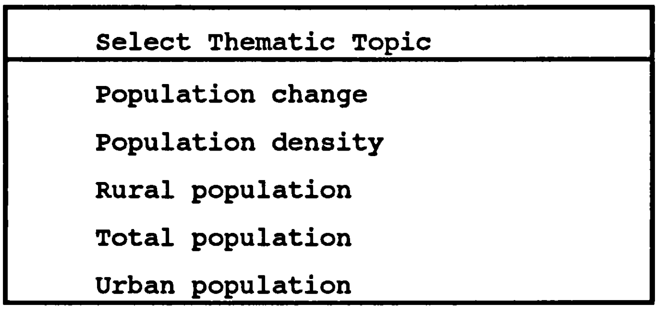
This checking procedure is seen to be of particular value when user selection is implemented where any omissions found would be presented to the user. Clearly when selecting from a menu with many choices (currently 18 for base information) it would be easy for the user to accidentally omit a feature further up the hierarchy or generally required for plotting a selected feature (e.g. if colour fill for lakes is required then the lake shores must also be selected).

The list of base information classes to be included in the map are asserted into the frame as are each individual feature with its associated probability. The latter is done to facilitate base map editing at a later stage, although it is recognised that features could only be removed later with this approach. It would be possible to store the unselected features and their scores separately, but this is not considered necessary at present. To improve the flexibility at the modify stage a threshold could be used to store possible additional features which narrowly missed selection. This would at least remove categories of no value to the current map topic.

Theme_info_list

Three scenarios arise in selecting theme information. First, the list of theme information in 'map_content' (see above) may be empty, e.g. for a topographic map; second, there may be only one single theme, in which case this is asserted into the slot; or third there may be several possible sets of theme information for the selected topic or class.

For this third case, the user is presented with a menu listing the theme data sets available for that map topic (Figure 8.6). The user is asked to select the topic(s) to be included. The menu invoked here allows multiple selections to be made from it. If a single topic is selected this is added to the frame and the selection phase is completed. Alternatively, if multiple topics are selected from the menu, a recursive procedure is called to determine the priority order of the topics selected.



Select Thematic Topic

Population change
Population density
Rural population
Total population
Urban population

Figure 8.6
Example of thematic topic selection - Map_topic = Population

In this second phase, the first menu shows all those topics selected and the user is asked to indicate the most important topic. Once a topic has been picked, it is removed from the list shown in subsequent menus. When a null response is made to the menu of remaining topics or all have been selected, the topics are asserted into the frame as a list in priority order. The priority order is used to determine which topic will be assigned representation method and symbols first, resulting in it being assigned the preferred values. For example, a climatic map may show both precipitation and temperature. The preferred representation method for both of these is layer colours. As it is only possible to show one set of area colour symbols, one topic must use an alternative method (isolines). Only the user can determine which has priority.

Although perhaps something that should be considered at this stage, the selection of the number of classes and setting class intervals for numerical data is part of the symbolisation process, and similarly selecting any subset of classes from a topic with categorical data is handled after the representation method has been chosen for the class.

MODULE 'SYMBOLISATION'

This module is by far the largest part of the whole system and has gone through several major iterations in its development. The central part, the assigning of graphic variables to the selected features presents numerous problems, the two greatest being the almost infinite number of permutations possible and how to check for conflicts between them. Two main versions of this central section are discussed, the first being an early attempt using a simplified set of graphic variables and the second, the current implementation, using a more comprehensive set of graphic variables. Although the author is critical of several other proposed cartographic design expert systems for concentrating almost entirely upon symbolisation, it must be recognised that although the stages described above are important and can be of considerable help to the map author, it is this symbolisation aspect which is the most difficult to resolve in a satisfactory manner, largely due to the combinatorial expansion of the search space.

The general sequence of steps in this module is:

1. Get the list of selected features and order this list in terms of priority for showing features.
2. Work through this list assigning representation methods. Initially assign the preferred representation to the current item, then check this against those already assigned. If there is no clash then proceed to the next item on the list. If there is a clash, choose an alternative representation method. If no viable alternative is available then backtrack to the previous item.
3. Assign specific symbols to features / representations. Again check for conflicts.
4. Assign symbols to appropriate plot level (i.e. layer or stratum).

Due to the complexity of the inference mechanism and the memory requirements of the knowledge bases associated with this module, it is in fact composed of four separately compiled, or global, sub-modules. The initial symbolisation module (symbolisation) deals with assembling the contents and assigning the representation methods. The second module (getsymbols) assigns symbols to the features which are checked by calls to the third part (check_symbol) at each stage. Once all symbols have been assigned the final part (assign_levels) adds the level to each symbol specification. Finally the 'getsymbols' displays the symbol specifications. Both 'symbolisation' and 'getsymbols' are called by the predicate 'design_map', whereas 'check_symbol' and 'assign_levels' are both called by 'getsymbols'.

Assemble

It would be possible to simply add the list of base information to the theme information then assign symbols to features on the list, but several of the base information classes are actually hierarchically related, e.g. major rivers, large rivers, other rivers, and it is more logical to treat these as a single class of phenomena for assigning representation method. Thus, a first stage is to build a list of classes of base information to be included, such as roads, settlements, rivers, etc. This is achieved by using the 'member_of' fact, in the form:

```
member-of (Feature_class, Feature),
e.g. member_of("Rivers", "Major Rivers")
```

in the knowledge base. These higher level classes are then appended to the priority ordered theme information classes and asserted into the working knowledge base. This information about feature classes is not stored as part of the map specification frame, i.e. the base information is stored in the frame at the feature level.

Representation

Most of the processing up to this point has involved forward chaining and could equally well have been carried out with a procedural programming language. Both representation and symbolism tasks make extensive use of the power of Prolog as a declarative language and operate in an essentially backward chaining paradigm. Both recursion and backtracking are essential components of finding the solution and generally a 'best first' search is followed.

The main predicate for assigning representations is 'assign_reps' which has three arguments, each of which are lists:

```
assign_reps ([features to be processed],[reps assigned so far],
                   [final reps])
```

This predicate is recursive, which means that the main clause works on the input list of features by repeatedly calling itself to process the remainder of the list. Wherever possible, for efficient memory management, recursive predicates should contain a 'cut' (ideally immediately before the last predicate) to prevent having to retain backtracking points in every recursive call, and should also make the recursive call to itself the last predicate in the rule (known as tail elimination). However, if it is necessary to process a list and retain the ability of backtracking to earlier parts of the list (as is required here), then backtracking points must be retained, with their necessary drain on memory, i.e. a pointer must be stored in memory for every backtracking point. Only when program execution passes a cut are the stored backtracking pointers removed, freeing memory.

Possible representations for each class of information are derived from the nature of the phenomenon and the spatial and attribute data as described in Chapter 5,[3] and are stored in the knowledge base in the form:

```
representation_type ( feature class , representation type,
                                probability )
e.g.
representation_type ( "Settlements" , "ranked points", 10)
```

Starting with the first feature class on the list, the preferred representation type is found and added to the list of representations assigned so far. The same procedure is followed for the subsequent feature classes, except that before adding it to the list a check is made to see if it conflicts with those already on the list of representations. Conflicts between representation methods are stored as facts in the knowledge base in the form:

```
conflict ( context, value 1, value 2 )
e.g.
conflict ( "rep type", "categorical - one level",
              "layers - unipolar" ).
```

In order that each conflict need only be entered into the knowledge base once, the check for conflicts is made in both the forward and reverse order for all representation types already selected (i.e. check_rep is recursive). For example, checking for conflicts with the current representation type 'Rep' and the list of previously selected rep_types '[First | Rest]':

```
check_rep ( _ , []).      % end of list of selected reps
                          % no conflicts.
check_rep ( Rep, [First | Rest]):-        % main clause
  not (conflict ( "rep_type", Rep, First )),
  not (conflict ( "rep_type", First, Rep )),
  check_rep ( Rep, Rest ).      % recursive call to check
                                % against rest of reps already
                                % assigned
```

Note that there need to be two clauses for this predicate. The second clause is the main one that does the actual checking, whereas the first clause

[3] Consideration has been given to automating this function based on meta-knowledge about the features in the database (see Chapter 7) but this is currently seen as a diversion from the main aims of the project.

merely checks to see if the end of the list has been reached, [ ] signifying an empty list. This type of structure is always required with recursive rules so that they know when recursion has finished. In some of the subsequent program extracts this first clause is omitted for clarity.

If there is a conflict, 'check_rep' fails causing a backtrack to the next possible representation for the current feature class. If an alternative is found, this is again tested and so on. If no suitable alternative representation can be found then the 'assign_reps' rule fails and backtracks to the previous recursion of the rule. Here alternative solutions will be sought for the representation of that class. If another choice is found, the system will move forward again in the list of feature classes, if not it will backtrack further until an alternative solution can be found for a previously set feature class.

Although adopting a best first solution, the procedure adopted allows full traversing of the 'solution tree'. Further heuristics could be applied to limit the range of search, but the fact that most features can only have a limited number of representation types (often only one) limits the expansion of the search tree and the fact that the representations are chosen in preferential order means that little further optimisation would be achieved in practice. Theoretically if it proved impossible to find a complete set of representation methods the system should backtrack to the selection process and reduce the number of feature classes to be included. Practically, largely due to the nature of modular programming in PDC Prolog mentioned above, this cannot be done by the current system. To date in testing the system it has never failed to assign representations to all feature classes.

Once the complete list of representations has been determined the representation method for each feature class is asserted into the current map frame.

Symbolism

This stage proved to be particularly problematic in developing the system and numerous partial implementations were developed on paper. An early attempt at programming this section using a simplified set of graphic variables became overly complex and failed to work. After re-examining the functional specification presented in Chapter 6 a revised version, initially more elaborate in concept, but with a more fully developed set of graphic variables proved to be successful. Both versions are described below.

First attempt at symbolisation

In an effort to keep the program simple, the first attempt at the symbolisation stage involved using only three graphic variables, form, dimension and colour. The basic procedure was to get the first 'feature' from the list of contents and extract its representation from the frame and assign this an appropriate form, dimension and colour. An immediate difficulty here is that some 'features' will be represented by a number of identical symbols (e.g. all minor rivers will have the same form, dimension and colour), but other 'features' would be depicted by a number of different symbols (e.g. graduated points would all be the same form and colour, but there would be a number of different sizes of symbol). This difference was to be handled by noting that a feature could be complex (i.e. use a number of symbols). As discussed below, a more rigorous definition of 'feature classes' and features was used in the later attempt.

The selected graphic variables were then checked against lists of forms, dimensions and colours already used. If there were no clashes the values for the current feature were saved, the graphic variables added to the list of values used and then the next feature processed. If there was a clash, different symbols for the current feature were sought.

The basic predicate specifying each symbol took the form:

```
symb ( feature, representation, type, colour, dimension, form )
 e.g.
symb ( minor_river, network_branching, line, darkblue,
              norm_width, continuous)
```

The graphic variables used were closely linked to the values available in the Borland Graphics Interface of the PDC Prolog system (see discussion under Display Module) and all were assigned integer values (e.g. blue = 1, square = 3, dashed line = 2, etc.). Using these values clearly put considerable limitations on the system for further development, although it was thought that it would initially simplify the translation from graphic variables to actual display on the screen.

In order to assign the graphic variables, a series of rules were developed. Each rule was given a unique five digit number. The first digit was always 1; the second digit indicated the dimension of the symbol (0, 1, 2, 3) with 4 used for general rules; the third digit referred to the particular or main representation type the rule refers to (0 for general rules for that dimension); and the final two digits identified the individual rule.

Generally each rule dealt with a single graphic variable, but some set more than one. For example, rule 10101 (i.e. point feature, representation type 1, rule 01) set the symbol shape to be a dot:

```
rule (10101):-                     % dot shape
 assign ( form (dot) ).
```

Rule 10103 sets the colour of a point symbol:

```
rule (10103):-                     % dot colour
 output_media ( media ),
 background_colour ( media, bkcolour ),
 palette ( media, bkcolour, [ list of colours ] ),
                                   % get available colours
 menu ( "choose colour", [ list of colours ], Choice ),
 assign ( colour (Choice) ).
```

Rule 10301 (point feature, representation type 3, rule 01) sets both the size and shape of ranked point symbols:

```
rule (10301):-                     % size & shape
 current_feature ("Settlements"),  % known
 assign( symb("National Capital", "ranked points", point,
             null, 4, square) ),   % size given in pixels
 assign( symb("State Capital", "ranked points", point,
             null, 4, circle ) ),
 assign( symb("Major Town", "ranked points", point,
             null, 2, dot) ).
```

This rule is obviously specific to settlements. Another similar rule would be required to deal with other similar representations. Colour would be chosen by a rule similar to rule 10103, or indeed that rule could be used, replacing the 'null' value for colour.

This system operated by working through the ordered list of contents and calling up the list of rules that applied to each feature:

```
symbol_rules(feature_class, representation_type, list_of_rules)
```

```
e.g.
symbol_rules(settlements, "ranked points",
                    [10301,10302,10303]).
```

Rules for either the specific feature or the representation can be found by leaving the other value as the anonymous variable (_).

Overall processing was carried out by the recursive predicate get_symbols:

```
get_symbols(
       [Feature | Rest],       % current feature & remainder
       [Incolours],            % list of colours already used
       [Outcolours],           % colours passed forward
       [Indims],               % list of sizes used
       [Outdims],              % sizes passed forward
       [Informs],              % list of forms used
       [Outforms] )            % forms passed forward
```

The main clause for get_symbols was:

```
get_symbols ( [Feature|Rest], Incolours, Outcolours, Indims,
       Outdims, Informs, Outforms ):-
 frepresentation ( Feature, Rep_type ), % get rep type
 assert ( current_feat ( Feature )),   % these are placed in
 assert ( current_rep ( Rep_type )),   % a working database
 symbol_rules ( Feature , Rep_type, [Rule_numbers] ),
                                     % get list of rules
 apply_rules ( [Rule_numbers] ),
                               % predicate that calls rules
 check_symbol,                 % predicate to check for clashes
                               % if OK go on, else look for
                               %     different symbols
 update_lists ( Incolours, Workcolours, Indims, Workdims,
                    Informs, Workforms ),
                               % adds new symbols to lists
 get_symbols ( Rest, Workcolours, Outcolours, Workdims,
       Outdims, Workforms, Outforms ).     % recursive call
```

There were several problems with this approach. While the idea of numbered rules and calling up the list of relevant rules in a given situation is a tried and proved method in classification type expert systems[4], it is rather restrictive in this application. All the rules in the list must succeed if the predicate is to succeed; there is no easy way of accepting partial success. Alternative lists of rules could be provided to allow for backtracking or for failure of the first set, but it may only be one rule on the list that needs to set a different value, not every rule in the list. This problem could be solved by having a list of lists, the rules in the sub-lists being alternatives, but it would still be very difficult to track attempts at symbolising features.

Secondly, as already noted, this approach did not seem to deal effectively with the different nature of representation types. In some cases all symbols will be the same for the whole feature class (or entity class), in some cases one graphic variable will differ for the symbols representing different features (entities) within a feature class and in other cases quite different symbols will be required for each feature. These differences were not elegantly dealt with by this method. There were also some practical difficulties in allowing the incorporation of both general rules for the representation type and rules for a specific feature class using a representation type.

The third and perhaps most critical limitation of this approach was the difficulty in expressing symbol differences and checking for conflicts between symbols. Limiting the graphic variables to three and having these expressed in relatively abstract values reflected by the colours, shapes and sizes allowable in the Borland Graphics Interface restricted flexibility and made any rules for finding conflicts cumbersome. It was also not possible to incorporate more complex symbols, such as cased roads, point symbols outlined in one colour and filled in another, etc.

Second attempt at symbolisation

Having decided that the simplistic approach initially adopted was not satisfactory, a further examination of what was actually required in terms of symbol specification was carried out. Perhaps not surprisingly, a more comprehensive set of graphic variables was developed. This does not exactly mirror the graphic variables discussed in most treaties on symbolisation (e.g. Bertin, 1967; Robinson et al, 1984; MacEachren, 1994) but is closer to the general cartographic approach than

[4] It has been suggested that this 'numbered rule' approach is one way of overcoming the inability of PDC/Turbo Prolog to assert clauses other than simple facts at run time (as can be done by some versions of Prolog). All the possible rules need to be pre-defined: those to be used are 'asserted' by asserting the rule number into the database at run time.

the model described above. The previous domain for 'symbolspec' was replaced by a more comprehensive domain which contained the following fields:

```
symbolspec(
  symbol type,       % point, line, or area
  hue,               % descriptive name
  lightness,         % descriptive name
  saturation,        % descriptive name
  form_code,         % class of form
  form,              % actual form name
  orientation,       % angle clockwise from north
  dimension,         % descriptive name
  level)             % integer value of level to be plotted on
                     % (i.e. a layered model is used similar to
                          % many CAD and mapping systems
```

This use of customised 'domains' or what may be referred to as 'types' in other languages is an important feature of PDC Prolog (but not all versions of Prolog). Domains perform three useful functions. First they allow meaningful names to be used in the declaration of predicates (e.g. the 'key' domain includes the codes for all the characters on the keyboard); second, they allow data structures to be declared that can be quite complex, but can be referred to simply - such as the symbolspec domain. Such complex domains can be referred to as a single variable, and can be passed by predicates in this way, or they can be referred to by the domain name followed by the components in brackets. E.g.:

```
symbol_a = symbolspec( point, blue, dark, high, geometric,
                      circle, 0, small, 301).
```

could be a valid statement. Symbol_a could then be passed to other statements in this shorthand form, and picked up by them either as a single entity or expanded to access the individual elements.

A list domain 'symbolspeclist' was also declared as a list of symbol specifications. This means that the complete set of symbol specifications for a feature class can be referred to as a single object, useful in simplifying programming and making the code more readable. But there is still the ability to refer to an individual symbol or an individual graphic variable.

In this second version no reference to the actual feature was included in the symbol specification domain, it containing only graphic variables and the plot level.

The feature referred to is maintained by the frame slot predicate 'fsymbolism' and can also be found in specifically defined symbols in the knowledge base in the 'ksymbolism' predicate, both of which take the form:

```
        symbolism( feature_class, feature, symbolspec).
```

where 'feature_class' is the main class of the feature and 'feature' is the specific entity. Both 'feature_class' and 'feature' will be the same where there are no subclasses. This allows a more elegant treatment of subclasses than in the first attempt. Complex (multi_part) symbols are dealt with by the use of appropriate values for 'form' and 'form_code' and look-up tables, or alternatively by having symbol components on different levels (i.e. two or more 'symbols' can be specified for the same feature). Thus, a cased road could have the casing on one level and the fill on a higher level.

As can be seen from the definition of 'symbolspec' above, most of the values use descriptive names rather than integer values as used previously. This simplifies the creation of menus when the user is asked to choose a value (a colour of 5 is meaningless to most users; in VGA display terms it is magenta). This approach keeps the specification of symbols more neutral to the eventual display media, with a series of look-up tables for the selected output device being used to translate from these generic terms to specific values for plotting. Table 8.1 lists the values currently available for the different variables.

The current method of assigning symbols follows similar principles to the first attempt, but with differences in implementation. The five steps involved are:

1) check to see of any symbols have been pre-specified;
2) use special rules for features if present;
3) use rules for representation type;
4) check for clashes, if none go on, else backtrack;
5) add symbol specifications to frame.

Table 8.1

Possible values for symbolspec variables

Variable	**Values**
Hue	purple, magenta, red, orange, yellow, green, cyan, blue, brown, grey, black, white
Lightness	pale, light, mid, dark [5]
Saturation / Brightness	low, mid, high
Form code -	
Type = area	solid, tint, pattern
Type = line	continuous, dotted, dashed, cased, complex
Type = point	geometric, combined, pictorial, subdivided
Form - (examples)	
F code = pattern	a range of patterns set in a look-up table
F code = dashed	short_dash, long_dash, dash_dot
F code = geometric	circle, dot, square, box, cross, plus
Dimension -	
Type = line	fine, medium, thick
Type = point	pixel, v_small, small, medium, large, v_large (note: these sizes only refer to nominal & ranked point symbols. Quantitative point symbols use separate rules to determine sizes)

The 'get_known' predicate is recursive and works through the ordered list of contents checking to see if any symbols are already specified. There are four possibilities, each dealt with by a 'symbol_known' rule. In the first case there is only one feature in the feature class (i.e. 'feature_class' and 'feature' are the same) and this is already specified in the frame. The second case is similar, except that a specific symbol for the feature is stored in the knowledge base. This assumes that the feature must always have that symbol and that there are no alternatives. This is important where there are widely recognised standard symbols for features, and

[5] An expansion to 6 or 7 terms would be useful, but it is difficult to find appropriate terms. For graduated colour series (e.g. on choropleth maps) it is still possible to use 5 or 6 tints of one hue.

also allows those setting up the knowledge base to impose standards or a 'house style' for certain features. The third and fourth clauses for 'symbol_known' are similar to the pair above, but deal with situations where there are a number of features in the feature class. Currently it is assumed that if one feature in the class is specified they all are, although this is a rather limited view and should be modified to allow only selected features to be pre-specified. For example, we might have standard symbols for National and State capitals, but allow flexibility in the symbols for other settlement classes. Like the 'symbol_rule' clause, the 'get_known' clauses check to see if there are any conflict between symbols, although it is not expected that clashes will be found.

The main clause of the predicate for assigning symbols, get_symbols1 is relatively short, and is again recursive, working through the list of contents:

```
( [ Feature_class | Rest ] ):-
frepresentation ( Feature_class, Rep ),     % get rep method
!,                       % cut forces backtrack to previous
                         % feature_class on failure of
                         % symbol_rule
symbol_rule ( Rule, Rep, Feature_class, Features, Symbols ),
                         % the predicate that gets the symbols
update_symbols( Feature_class, Features, Symbols ),
                               % adds symbol specs to frame
get_symbols1( Rest ).          % recursive call
```

Before examining the mechanism of the symbolisation rules it is worth describing how the symbol specifications are updated in the frame. The functioning of 'update_symbols' predicate is important as it must be able to retract previously asserted symbols on backtrack as well as adding new ones while processing is moving forwards in the list of feature classes. To achieve this it has an additional clause which is not called during the process of adding symbols to the frame, but is called during backtrack as Prolog looks for an alternative solution to the predicate. In fact this additional clause having removed the symbols from the frame then fails, causing the system to backtrack further to the symbol_rule predicate to seek alternative symbols for the feature_class whose specifications have just been retracted. The three clauses for this predicate are:

```
update_symbols ( _, _, []) .          % end of adding list of
                                      % symbols to frame

update_symbols ( Feature_class, [ Feature | Rest ],
                  [ Symbol | Symbols ] ):-
                             % normal case - assert new symbols
 assert ( fsymbolism ( Feature_class, Feature, Symbol ) ),
 update_symbols ( Feature_class, Rest, Symbols ).
                                      % recursive call

update_symbol ( Feature_class, _, _ ) :-
                                      % alternative on backtrack
 retractall ( fsymbolism ( Feature_class, _, _ ) ),
 !, fail.               % ensure fail to continue backtracking
```

Symbolisation rules take the general form:

```
symbol_rule ( number, representation_type, feature_class,
              [features], [symbol specifications] )
e.g.
symbol_rule(105,"ranked points","Settlements",Features,Symbols)
```

The number in symbol_rule is arbitrary and is used for record purposes, although generally it conforms to the rule numbering system described above. It could in future be used to help develop the explanation facility by recording the order of rules processed. The representation type is, with one exception, always required to be specified in the predicate call. The feature class is also always required. The lists of features and of symbols are returned by the predicate.

There are three main groups of symbol rules. The first group of rules, the only ones not requiring the representation type to be specified are used to check to see if the symbol for a feature should be the same, in whole or in part, as another symbol already specified. This utilises the knowledge base fact 'associate_with' in the form:

```
associate_with ( Feature_class 1, Feature_class 2,
       graphic variable, probability )
e.g.
associate_with ( "Seas", "Lake_fill", hue, 8 ).
```

This means that if the symbol for Seas has already been specified then there is an 80% probability that the hue for lake fill will be the same. In order to save replication, it is assumed that the reverse is also the case. Probabilities are included in the facts for completeness, but as only cases where there is a high probability of association are included in the knowledge base they are not utilised in the current implementation.

The second group of rules are those for particular feature classes. These are special versions of the general rules for the representation type reflecting some special treatment required in these cases. For example, while it would be possible to use the standard rules for "ranked points" to symbolise settlements, the set of symbols used to show the administrative status of settlements on small scale atlas maps is often different to that which would be used for other ordinally scaled point symbols.

The main group of rules are those for each representation type, there being at least one rule for each type. These rules vary in structure and complexity depending upon the representation type. In some cases fully specified symbols or sets of symbols will be extracted from the knowledge base. In others the individual graphic variables will be set in turn. For example, in setting area symbols for an 'unclassed areas' representation (political map) there are several sets of visually equivalent area colours stored in the knowledge base. These sets of symbols (or single graphic variables in other cases) may contain more symbols than is required by the current specification. It is, however, important that wherever possible these sets of symbols exceed the number likely to be required in order to allow for one or more of them to be eliminated due to conflict with other symbols on the map.

Generally the symbol_rules call a small set of rules to assign the actual graphic variables or symbols to the individual features, such rules including: assign_ranked_sizes; assign_line_colour, assign_tints; etc. While conceptually it would be preferable to have a single predicate called assign_symbol with multiple clauses, in practice due to the different information that has to be passed and returned in different situations it is simpler to have separate predicates.

To illustrate the processing involved, the rules for the representation type 'unclassed areas - one level' (2Ba in Figure 5.2) and for the actual assignment of available symbols to classes are given below.

```
PREDICATES (only main ones not mentioned before given here)
   assign_equiv_colours( % returns equally appearing symbols
                         % versions for points, line & areas
       SYMBOL,                 % symbol type - p,l,a
                               % or feature class
       STRINGLIST,             % Groups,
       SYMBOLSPECLIST)         % Groupsymbols
   assign_symbols(       % assigns fully specified symbols to
                         % features
       STRINGLIST,             % list of features to be done
                               %     - stops when list empty
       SYMBOLSPECLIST,         % symbols available for use
       SYMBOLSPECLIST,         % working symbols list
       SYMBOLSPECLIST)         % assigned symbols output
   ask_symbolset(
       SYMBOL,                 % Feature_class
       SYMBOL,                 % type of symbol set
       SYMBOL)                 % selected set

CLAUSES  % note: bold used to indicate the start of new clause
symbol_rule(22101,"unclassed areas - one level", Feature_class,
             Groups, Symbols):-       % 1 group for each symbol
   message("a minimum of four colours is required  ",
 "the system is currently limited to the possibilities shown"),
   menu(["  4","  5","  6"], " Select number of colours ",
             2,Choice),        % ask user how many colours
   Num = Choice + 3,           % add 3 to selected list pos'n
   str_int(N,Num),             % convert to string
   concat("group_names",N,Name),     % make up reference name
                               % for list of group names
   kstringlist(Name,Groups),         % get list of group names
                                     % from knowledge base
                               % one symbol will be assigned to
                               % each group name & stored
   !,                          % don't backtrack
   assign_equiv_colours(area,Groups,Symbols).
                               % call predicate to assign cols.
```

```
assign_equiv_colours(point,Features,Symbols):-    % first clause
                    % not used here as Feature_class <> point
    !,
    ksymbolset(equiv_point_colours,Symbollist),
    assign_symbols(Features,Symbollist,[],Symbols).
assign_equiv_colours(line,Features,Symbols):-     % second cl.
                    % not used here as Feature_class <> line
    !,
    ksymbolset(equiv_line_colours,Symbollist),
    assign_symbols(Features,Symbollist,[],Symbols).
assign_equiv_colours(area,Features,Symbols):-
                                        %third clause
                                  % in this case Features = Groups
    ask_symbolset(_,equiv_area_colours,Set),
                            % ask user which set of symbols to use
    ksymbolset(equiv_area_colours,Set,_,Symbollist),
                            % get set of symbol possibilities
    !,
    assign_symbols(Features,Symbollist,[],Symbols).
```

There are three clauses for assign_equivalent_colours: one for points, one for lines; and one for areas, the latter used in the example. This uses ask_symbolset to get the user's choice of symbol set, the options currently included being mid tones, light tones and light, bright colours. Each of these sets stored in the knowledge base contain at least seven different area colours which are approximately visually equivalent. Other sets could easily be added to the knowledge base. (Currently there is only one set of colours for points and lines, hence ask_symbolset is not included in the clauses for these.) Ask_symbolset uses the findall predicate to collect all symbol sets with the name passed to it in 'Type' - in this case 'equiv_area_colours' and presents these as a menu to the user.

```
ask_symbolset(Feature_class,Type,Set):-
                                        % choose a set of symbols
    findall(Name,ksymbolset(Type,Name,_,_),Names),
                            % find all sets for type in kbase
    menu(Names, " Select set of symbols ",1,Choice),
    member_from_index(Names,Choice,Set).    % get name of
                                        % selected set using index
                                        % number from menu
```

```
assign_symbols([],_,Symbols,Symbols):-!.   % all features done
assign_symbols(_,[],So_far,So_far):-       % clause when list
                                     % symbols finished before
                                     % all features assigned
   !,              % prevent search for alternative clause
   ermessage("insufficient symbols in set",
             "some features not symbolised"),
   fail.           % backtrack to previous feature_class
assign_symbols([_|Features],[Symbol|Symbollist],
             Insymbols,Outsymbols):-       % main clause
   check_symbol(Symbol),             %if ok continue, else fail
                                     %  and find next symbol
   !,
   append(Insymbols,[Symbol],Working), % add symbol to list
   assign_symbols(Features,Symbollist,Working,Outsymbols).
                               % next feature - recursive call
assign_symbols(Features,[_|Symbollist],Insymbols,Outsymbols):-
                               % after check_symbol fail
                               % - try next symbol in list
   assign_symbols(Features,Symbollist,Insymbols,Outsymbols).
```

Assign_symbols is the predicate which does the work of assigning an allowable symbol to the right number of groups. This is a recursive predicate with four different clauses. The first clause simply checks to see if all groups have been assigned a symbol and the task completed. The second clause checks for the list of symbols being exhausted before all groups have been assigned. If this happens the system backtracks and looks for alternative symbols to previous features, hopefully freeing further possibilities for this feature_class. The third clause is the normal processing clause. It first checks that the next symbol on the list has no conflicts. If it passes this test, the symbol is added to the list of symbols to use and the recursive call is used with the reminder of the list of features (groups) and symbols. If the check for conflicts fails, the third clause fails and the four clause is entered. Here the first symbol on the list (the one in conflict) is removed and the predicate called again with the new list.

Thus, it is important that the lists of equivalent colours have more than the number of features or groups to be symbolised if at all possible to allow for one or more symbols to be rejected without causing this whole predicate and hence the current symbol_rule to fail. The greater the number of equivalent colours on the list, the less likely failure is.

Finally, there is a symbol_rule intended only for testing which always succeeds if all others fail. This sets the symbol to null and reports this to the user. When rules for all eventualities have been developed this rule will be superfluous. It does however mean that the symbolisation process will never fail, but may not assign symbols to some features, although this only happens when all possibilities have been tested. Thus the symbolisation process uses heuristics to carry out a best first search for the solution, but the heuristics do not cut off the possibility of other branches of the search tree being followed eventually.

Wherever possible the ideals of knowledge based systems have been applied to the solution of the problem. For example, as discussed above, in choosing the set of colours for a political map, there are several possible sets of 'visually equivalent area colour' symbols known to the system. While it would be possible for the first of these to be selected or some priority to be established, in reality this is a case where only the user can decide which is most appropriate. Thus, in such situations the system will collect together the possibilities, construct a list of these and present this to the user in the form of a menu for a choice to be made. The default choice highlighted in the menu will either be the first case matching the current criteria found in the knowledge base, or will be the case with the highest level of confidence. The important principle is that the menus are not pre-defined in the inference engine; they are built during run time from information in the knowledge base to meet the current requirements.

Checking selected symbols.

An integral part of assigning symbols is checking that the symbol selected does not conflict with symbols already specified. This is without doubt the most complex problem to be solved by the system and was largely responsible for the failure of the first attempt at symbolisation. The main difficulty is in determining all the possible combinations of conflicts that may occur between symbol type (point, line, or area) and the graphic variables. This is one area where the human visual system is particularly efficient, but difficult to simulate, although the abundance of maps published with poorly differentiated symbols does lead to speculation about the designers' attention to this detail.

Essentially the problem is how much difference does there have to be for two symbols to be seen as being different, or discriminated from one another. Our ability to discriminate in not equal for all perceptual variables. In terms of the relatively small visual images involved here, generally our ability to discriminate is better for colour than it is for size, and shape is the poorest (e.g. see Forrest, 1981; Williams, 1971), although there are many influencing factors and situations which contradict this. Despite the large amount of perceptual testing of cartographic symbols carried out in the last thirty years there are still no easily available guides to this which can simply be entered into the system.

The current implementation uses four basic predicates: one which checks a fully specified symbol, one which checks colours; one for form; and one for dimension. The first two of these are the most important and most frequently used. Generally conflicts of the latter two types are less likely to arise given the current implementation of the system. The check_symbol predicate for complete symbols calls the other predicates in turn to check these variables.

The reason for having the ability to check the individual graphic variables as well as the whole symbol is to do with the order of specifying the graphic variables. Although there are exceptions, most symbol_rules assign colour first, then form then size. If there was only one point to check for the symbol being unsuitable then backtracking from the symbol_check rule would look for alternative graphic variables in the reverse order to which they were determined. While this could be exploited in some cases by changing the order in which the graphic variables are determined, there are cases where it is useful to determine that, for example, the colour is permissible before attempting to determine form and dimension. This is especially so where a set of symbols will all have the same colour, but vary in form and/or dimension.

It is not sufficient to simply check that all symbols are different by some standard threshold values. There are distinct differences in what is suitable for differentiating features within the same feature class as opposed to between two different feature classes. For example, in a layer coloured map all layers can have the same hue; the variation between the symbols for features is by lightness, and these lightness differences may be relatively small. Generally speaking different feature classes will have different hues; if they have the same hue, the differences in lightness and saturation must be significant.

The operation of check_symbol is to:

1. if the current feature has exactly the same symbol as a previously symbolised feature then fail.
2. if it has the same hue as an existing symbol then goto 4.
3. the colour is checked using a set of conflict rules to see that the colour is not too similar to the colour for a feature in another feature class. If conflict then goto 5 else goto 7
4. if lightness and saturation similar then fail else goto 7
5. If either 2 or 3 then check form - if same as existing then goto 6 else goto 7
6. if dimension similar then fail else goto 7
7. check_symbol succeeds - move on to next feature

This is clearly much more limited than the skill and visual processing employed by the expert cartographer, but the main aim is to eliminate potential conflicts and produce sensible symbols rather than ideal ones.

Levels

The final part of the symbolisation module assigns each feature class to a level for plotting. Before plotting commences the symbols will be sorted by level so that they are plotted in the correct order. This approach is similar to that of most CAD packages, drawing packages, etc., where the information is divided into a number of layers (explicit or not) and anything on the top layer will cover anything on lower layers (some systems do allow layers to be transparent rather than opaque).

The general sequence is to plot areas first, then lines, then points. Names would normally be plotted last. A simple approach to assigning layers is used by the system. Each representation method has a unique layer number associated with it which is simply found from the knowledge base. If two classes of features have the same representation method then they would both be assigned the same level. The order in which they would be plotted would be dependent on the order the feature class was selected and symbolised. Apart from combinations with isolated areas (which are always assigned the highest levels for areas), only one set of area symbols is permitted, so this approach should have few negative consequences, although it could perhaps result in point symbols being plotted partially overlapped. The concepts described by Mackaness (1984) could perhaps be incorporated at a later date to resolve such spatial conflicts.

There is one case where the general sequence of areas, then lines, then points is altered. Lakes are always plotted on a higher level than rivers. This means that river networks can be complete (i.e. a river segment can be digitised through lakes), allowing a complete drainage network to be shown without lakes being shown. This is particularly useful at smaller scales when lakes become too small to be shown. If lakes are selected, plotting them on top of the rivers covers the appropriate line segment without recourse to generalisation operators[6].

Finally, having assigned levels to all feature classes a list of all the levels used is constructed, sorted into numerical order (i.e. plotting order) and the list stored for use by the 'display' module.

DISPLAY

Having accomplished the preceding steps all slots in the frame are filled, the map specification is now complete and can be displayed or printed. Theoretically the process of transferring this specification to a graphical plot on the screen is relatively simple and generally would not be considered as part of the main expert system, but rather as a graphical output module. In using PDC Prolog for this module there are some difficulties that perhaps would not occur with other systems or languages. There are also some cases where applying the symbol specification to the data is not trivial, a particular case being in constructing a coloured political map. This case is dealt with in some detail following the general description of the operation of this module. It should be noted that the modular construction of the system would allow alternative display modules to be incorporated for other output devices.

The Display module is split into two main components. First, the main part which has largely a 'housekeeping' function and controls the initialisation of the graphics system, the opening of windows, the sending of messages to the graphics screen and, when plotting is complete, the closing down of the graphics system. It contains the procedures which set up the transformation parameters to convert real world co-ordinates into screen co-ordinates according to the selected map projection, map scale and display device. Also, the main part loads the metadata into memory to allow symbolic names for feature classes to be related to the correct file and feature names in the database.

[6] It is noticable that in The Times 'World Map & Database' that rivers are plotted on top of the lakes, the default colours being slightly different.

The second component of the Display module is the actual plotting program. It works level by level, converts the symbol specifications into the appropriate values for the graphics system, reads the data from the database and plots it on the screen. The main predicate is plot_level, which is recursive and takes the form:

```
plot_level ([Level | Rest]):-
   symbolism (F_class, _, symbolspec(_,_,_,_,_,_,_,Level)),
             % get feature class for this level from frame
   meta_data(F_class,_,_,_,_,_,Coordname,_,
             Datafile,Dataname),
                   % get names and files from metadata file
   coord_file(Coordname,_,_,_,_,_,_,Coordfile,_)
                   % find coordinate file
   assert (current_class (F_class)),  % store current class
   assert (coord_file (Coordfile)),   %     "     "   coordfile
   assert (data_file (Datafile)),     %     "     "   datafile
   assert (data_name (Dataname)),     %     "     "   data name
   representation (F_class, Rep),     % get representation type
                                      % from frame
   plot_class (F_class, Rep),         % call plotting predicate
   retractall (_, currentlevel)       % remove all names etc.
                                      %     before next level
   plot_level (Rest).           % recursive call to rest of list
```

There is a separate clause for the plot_class predicate for each of the representation types, although some of these are very similar. Essentially plot_class gets the symbol specifications for the features in the current feature_class from the frame, calls set_symbol to convert the symbolic names for the graphic variables into appropriate values for the graphics system, then calls the appropriate plotting routine (plot_points, plot_lines, or plot_areas). For feature classes where there are several features with different symbols, plot_class uses two clauses and fail to cause backtracking, rather than recursion, e.g.:

```
plot_class ( F_class, "ranked points") :-
   symbolism (F_class, Feature, symbolspec(_,Hue,Light,Sat,
       Form_code, Form, Orient, Dim, _)),        % from frame
   set_symbol ( point, Feature, Hue, Light, Sat, Form_code,
                   Form, Orient, Dim),
    fail.           % forces backtrack to get other Features
                    %     in class
plot_class ( _, "ranked points"):-
    plot_points (F_class).      % call plotting routines
```

Set_symbol uses lookup tables to convert the descriptive names for the graphic variables as stored in the frame into the specific values required for the Borland Graphics Interface (BGI) built into PDC Prolog, e.g. "blue" = 1, "thick" = 3, etc. The lookup tables also provide values for the fill_style and fill_pattern variables used by the BGI to fill areas. As a result of the limitation of the number of colours available with the BGI (sixteen), although the specification in the frame may be for a solid light blue colour, the conversion from the lookup table may give a pattern of a darker blue - in effect a tint. In fact the symbolisation module largely treats solids and tints the same, as in theory the dots in a tint are below the visual threshold, so that a tint should appear solid to the map user. Due to the resolution of the screen, in most cases 'tints' are made up of perceptible dots or patterns on the screen, but an attempt is made to kept this to the minimum. In producing maps on other output media a similar, but different, set of lookup tables would be used.

It is at this stage that complex symbols are split into their component parts. For example cased roads are stored in terms of a background symbol (the casing) and a foreground symbol. The BGI values for each symbol are stored in a database predicate 'setup', two examples of which, one simple, one compound, are shown in Figure 8.7.

The first of these examples is a cased line representing a highway by a wide red line with a thin yellow line superimposed upon it. The second example is for the light blue colour (a tint of blue) used to fill sea areas. This combination of variables allows for most relatively simple eventualities to be covered, although there is a limit to the complexity of individual symbols. The use of a fill colour as well as a main colour allows areas to be outlined in one colour and filled in another, or point symbols to be outlined. If this is not required, only the main colour is specified. The fill style for areas can be solid (value = 0), use the standard BGI patterns (1 - 11), most of which are not suitable for the purposes here because they are rather coarse, or if set to 12 make use of the fill pattern specified in "fill pattern". This fill pattern is defined as a list of hexadecimal values which determine which pixels are switched on or off in an 8x8 pattern cell. A set of suitable patterns for mapping were specifically developed for the system (see appendix C). In the second example above, the BGI pattern 9 is used, which is a close dot fill.

slot name:	e.g. 1	meaning	e.g.2	meaning
feature	"highway"		"seas"	
main draw colour	4	red	1	blue
line or point style	0	solid line	0	solid line
line pattern	0	no pattern	0	none
width or size	3	3 pixels	1	one pixel
fill colour or second line colour	14	yellow	0	none
fill style or second line style	0	solid line	9	close_dot_pattern
fill pattern	""	none	""	none
second width or size	1	1 pixel wide	0	none

note: line style must be set to 'solid' and size to 1 pixel for area fills to plot correctly

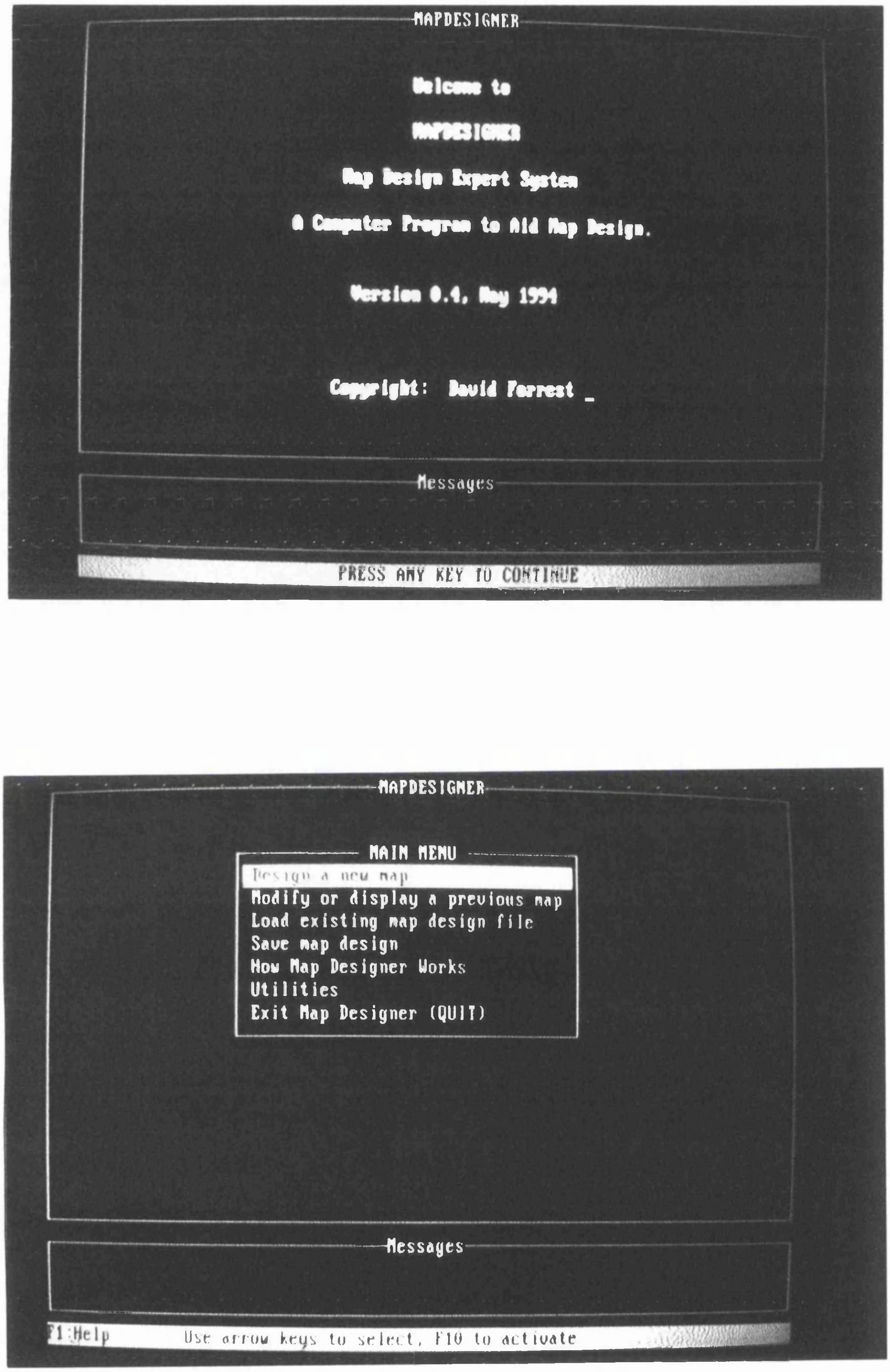
Figure 8.7

Examples of plotting parameters

A limitation of patterns in the BGI is that only one colour can be used. The background colour must be the same for the whole screen, and is normally set to white by MapDesigner. Some graphics systems allow area patterns to be defined by both a background colour and an overlaid pattern for each polygon, but this is not possible with the BGI. This does place some limitations on the options for area colouring, particularly if it is desirable to overlay one feature class by patterns over another feature class shown by colours.

Example run of MAPDESIGNER

Having described all the stages involved in producing a map with MapDesigner, a complete run of the system is shown in Figure 8.8 (a - r). This shows all the menus and output from the system in producing a Relief map of Nigeria. The associated map frame is shown in Figure 8.9. A Relief map with only a change in input values from overview to analysis is shown in Figure 8.10. Further examples of maps of different topics and scales and alternative menus used for different topics or representation methods are given in Appendix D

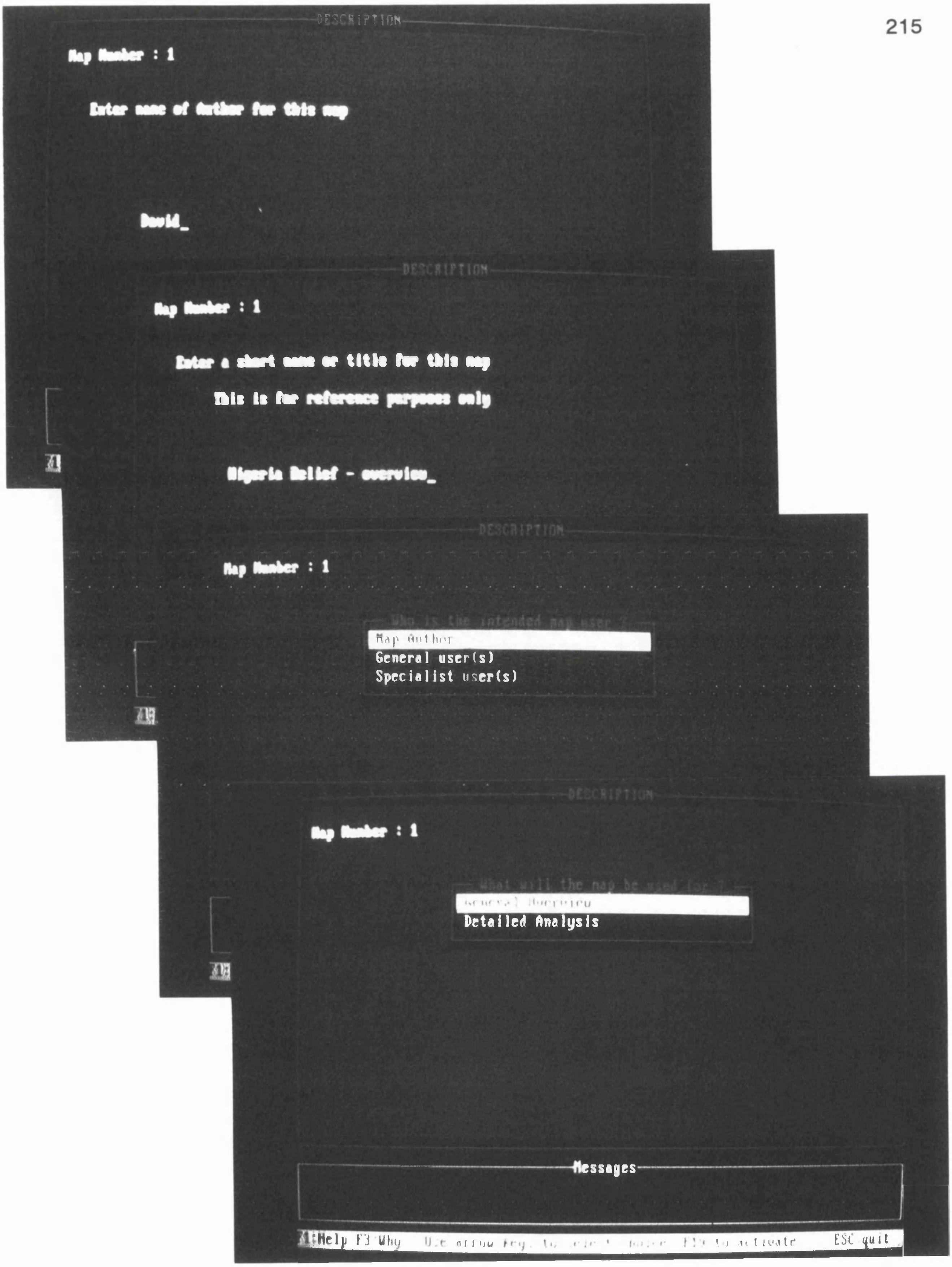


```
MAPDESIGNER

Welcome to

MAPDESIGNER

Map Design Expert System

A Computer Program to Aid Map Design.

Version 0.4, May 1994

Copyright:  David Forrest _

Messages

PRESS ANY KEY TO CONTINUE
```

```
MAPDESIGNER

MAIN MENU
Design a new map
Modify or display a previous map
Load existing map design file
Save map design
How Map Designer Works
Utilities
Exit Map Designer (QUIT)

Messages

F1:Help        Use arrow keys to select, F10 to activate
```

Figure 8.8 a & b

The Welcome screen and the main menus screen



DESCRIPTION

Map Number : 1

Enter name of Author for this map

David_

DESCRIPTION

Map Number : 1

Enter a short name or title for this map

This is for reference purposes only

Nigeria Relief - overview_

DESCRIPTION

Map Number : 1

Who is the intended map user ?

Map Author
General user(s)
Specialist user(s)

DESCRIPTION

Map Number : 1

What will the map be used for ?

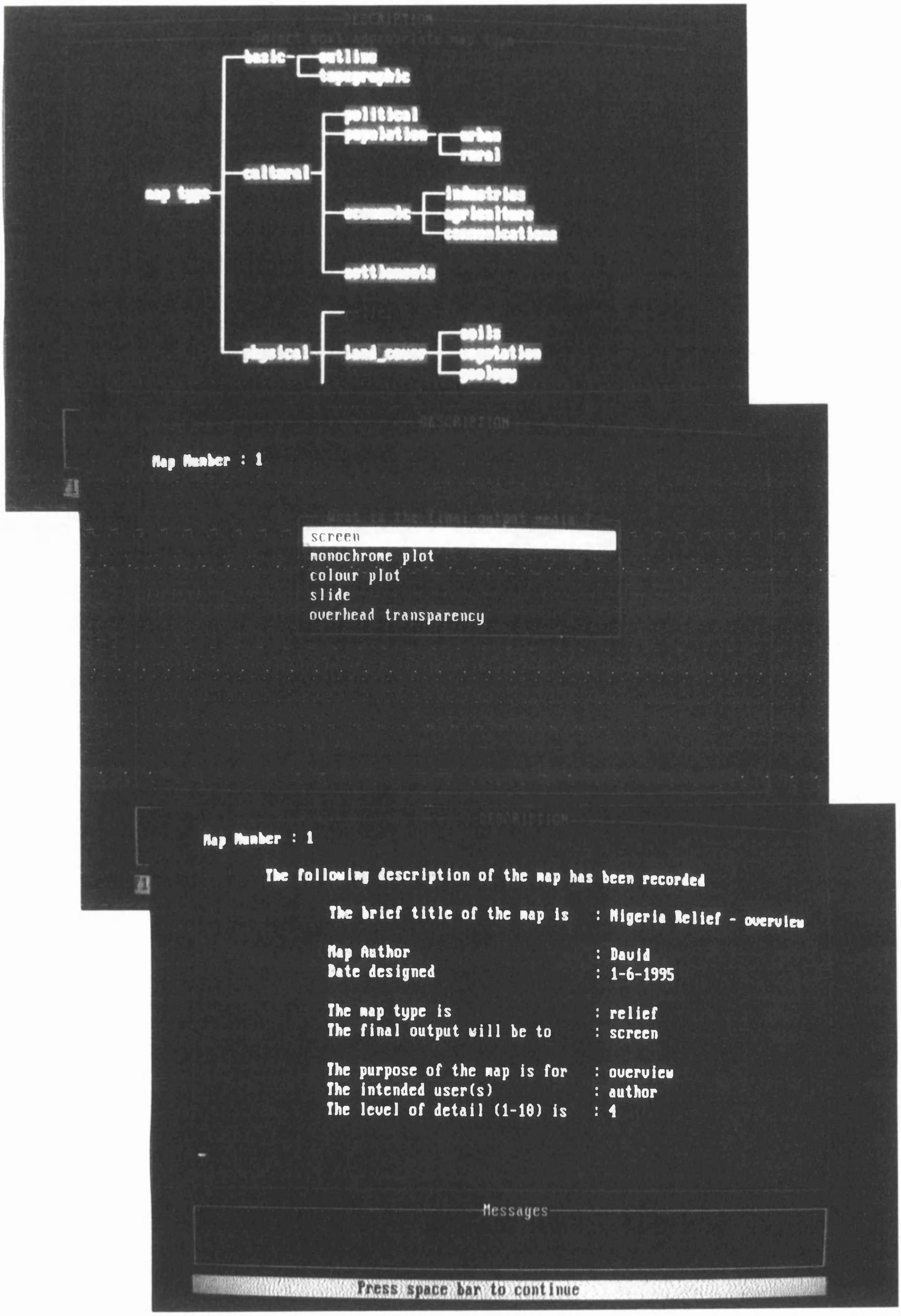
General Overview
Detailed Analysis

Messages

F1:Help F3:Why Use arrow keys to select choice F10 to activate ESC quit

Figure 8.8 c, d, e & f

The map_author, map_title, map_user and map_purpose screens



DESCRIPTION

Map Number : 1

What is the final output media ?

```
screen
monochrome plot
colour plot
slide
overhead transparency
```

DESCRIPTION

Map Number : 1

The following description of the map has been recorded

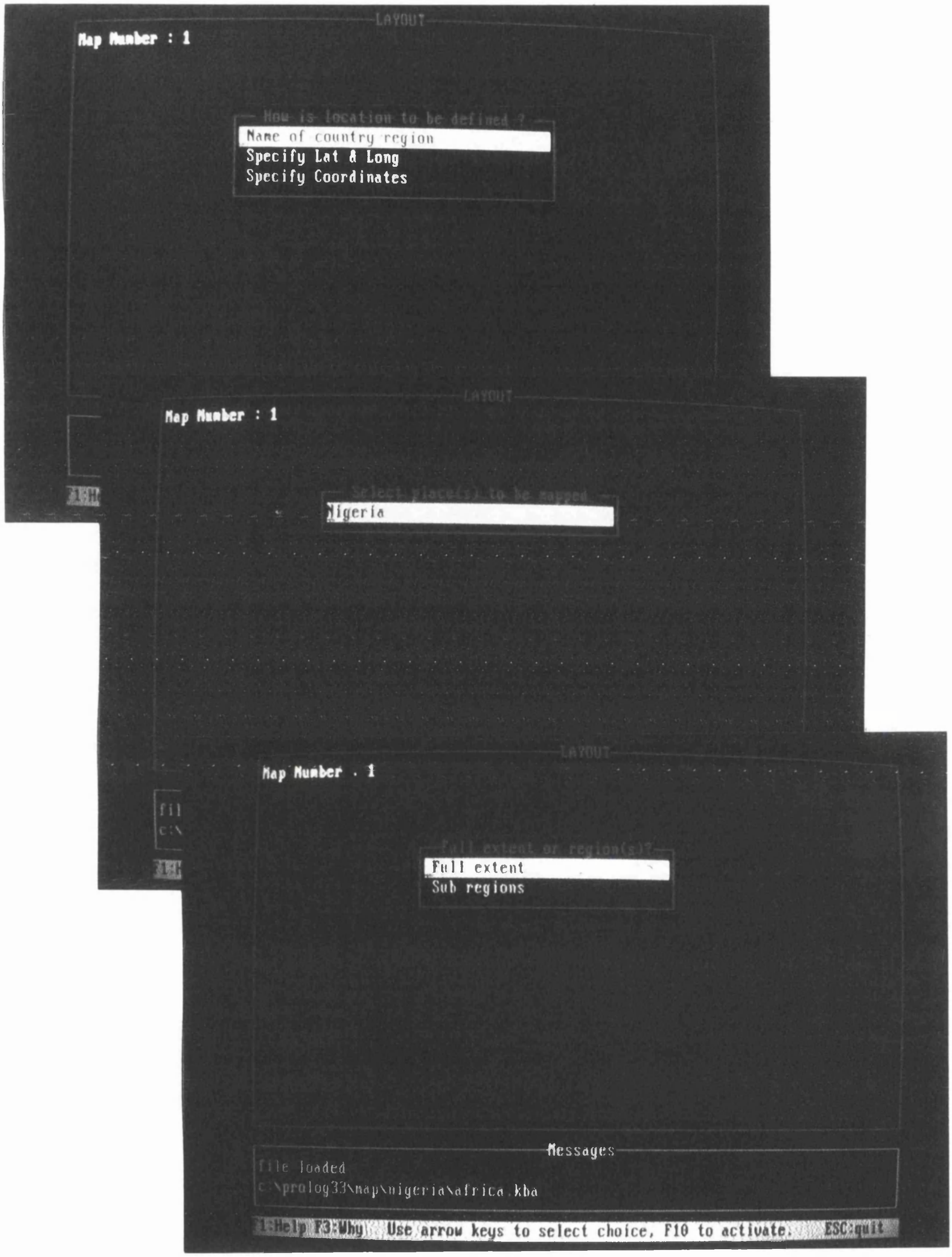
```
The brief title of the map is    : Nigeria Relief - overview

Map Author                       : David
Date designed                    : 1-6-1995

The map type is                  : relief
The final output will be to      : screen

The purpose of the map is for    : overview
The intended user(s)             : author
The level of detail (1-10) is    : 4
```

Messages

Press space bar to continue

Figure 8.8 g, h & i

The map_topic, output_media and show_description screens

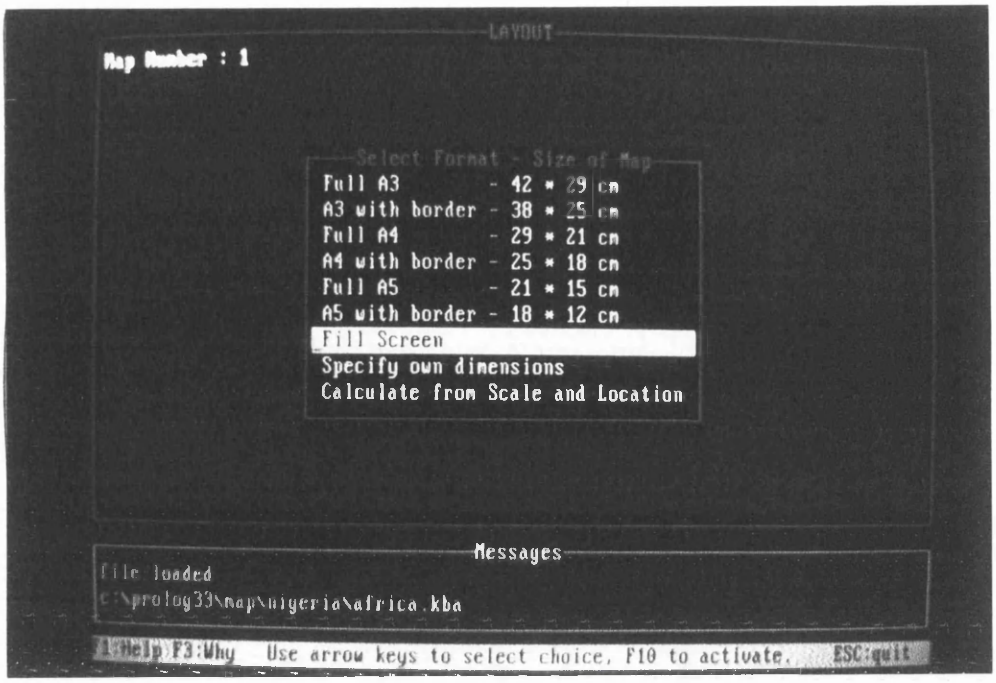


```
file loaded
c:\prolog33\map\nigeria\africa.kba
```

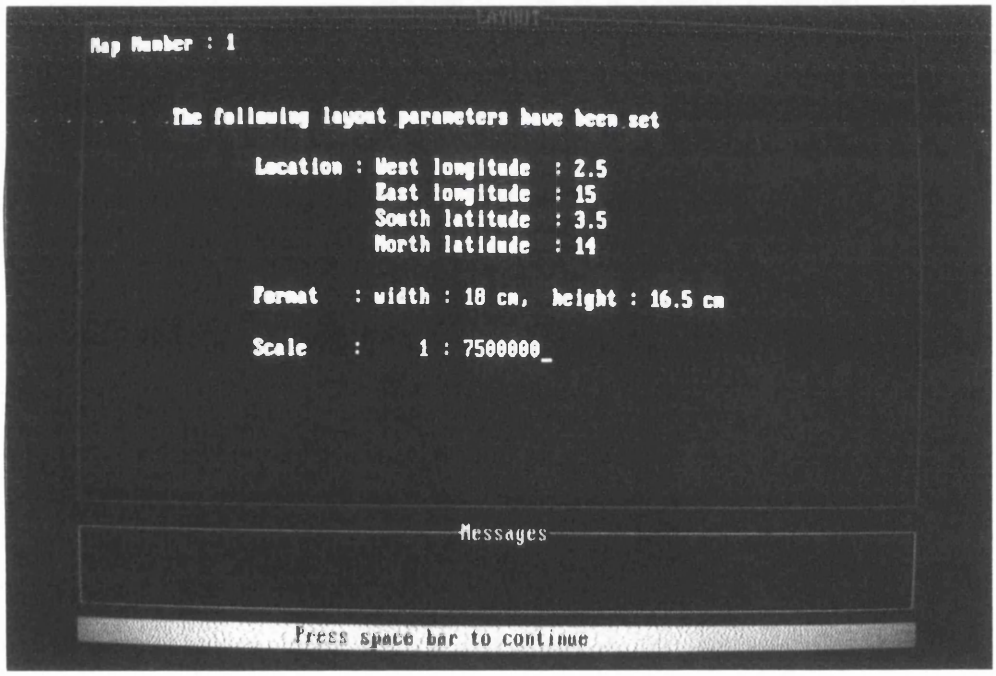
Use arrow keys to select choice, F10 to activate.

Figure 8.8 j, k & l

The location selection method, select by place and select place supplementary screens



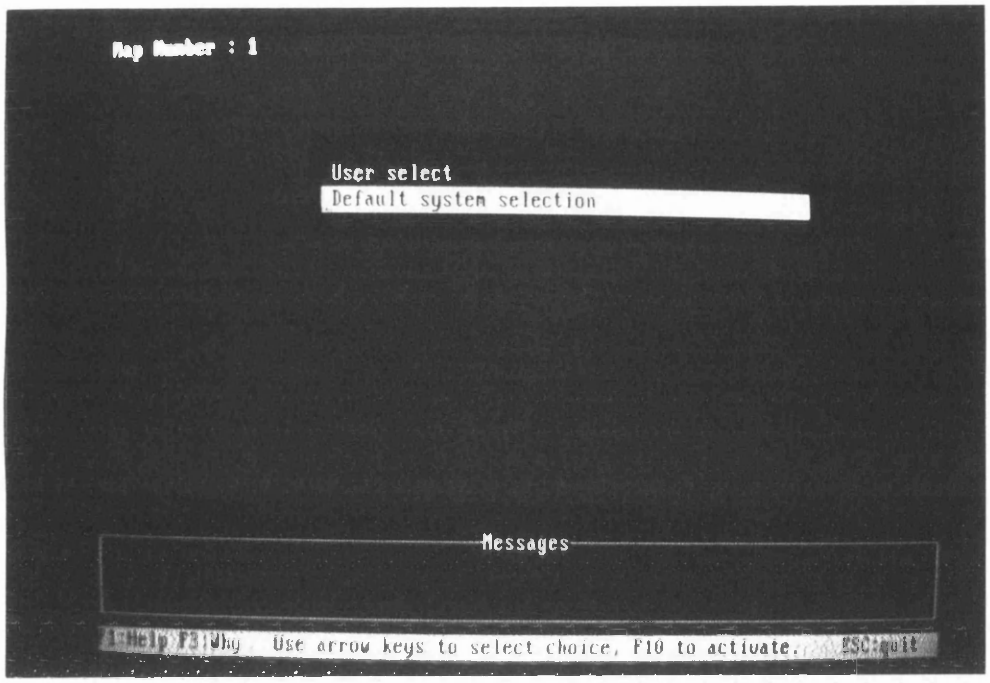
```
LAYOUT
Map Number : 1

Select Format - Size of Map
Full A3          - 42 * 29 cm
A3 with border   - 38 * 25 cm
Full A4          - 29 * 21 cm
A4 with border   - 25 * 18 cm
Full A5          - 21 * 15 cm
A5 with border   - 18 * 12 cm
Fill Screen
Specify own dimensions
Calculate from Scale and Location

Messages
File loaded
c:\prolog33\map\nigeria\africa.kba
F1:Help F3:Why   Use arrow keys to select choice, F10 to activate.   ESC:quit
```



```
LAYOUT
Map Number : 1

     The following layout parameters have been set

          Location : West longitude  : 2.5
                     East longitude  : 15
                     South latitude  : 3.5
                     North latidude  : 14

          Format   : width : 18 cm,  height : 16.5 cm

          Scale    :     1 : 7500000_

Messages
Press space bar to continue
```

Figure 8.8 m & n

The select format and show_layout screens

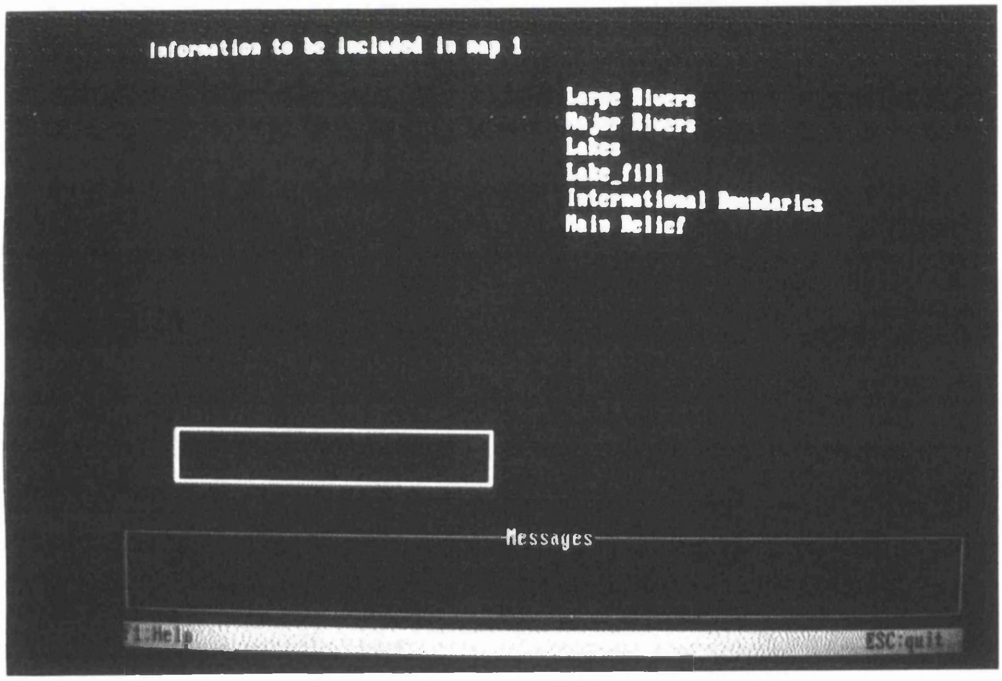


Map Number : 1

User select
Default system selection

Messages

1:Help F3:Why Use arrow keys to select choice, F10 to activate. ESC:quit



Information to be included in map 1

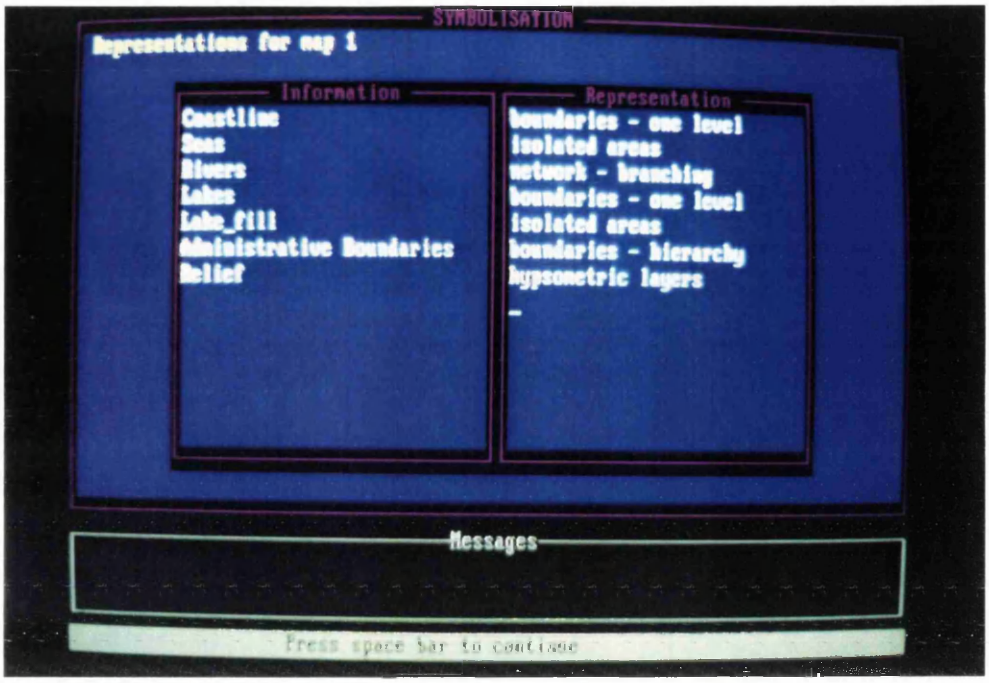
Large Rivers
Major Rivers
Lakes
Lake_fill
International Boundaries
Main Relief

Messages

1:Help ESC:quit

Figure 8.8 o & p

The data selection and show_selection screens



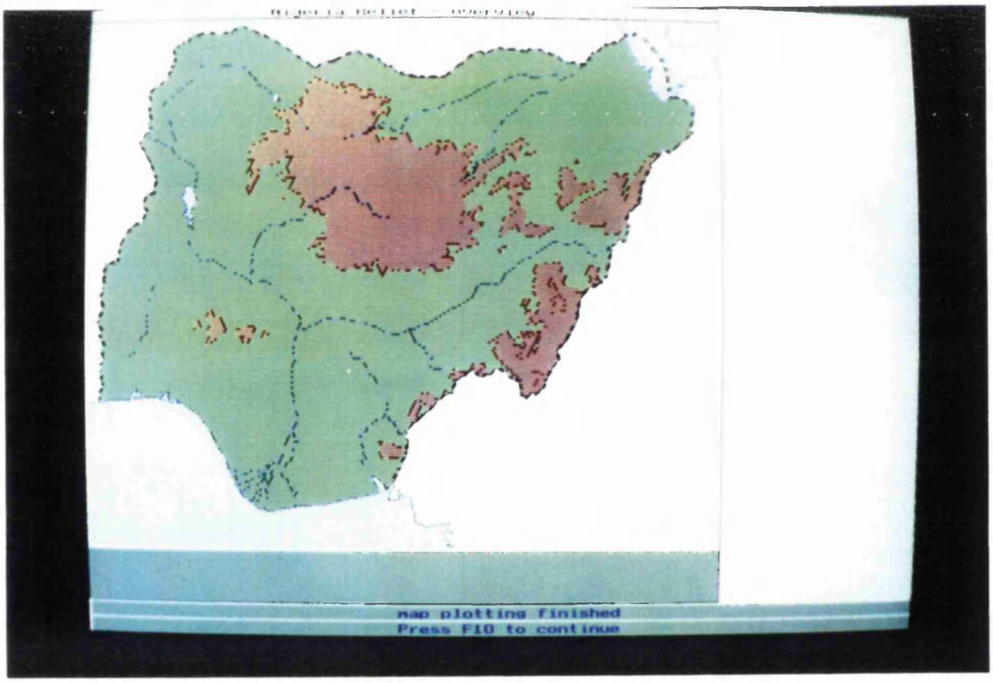
SYMBOLISATION

Representations for map 1

Information	Representation
Coastline	boundaries - one level
Seas	isolated areas
Rivers	network - branching
Lakes	boundaries - one level
Lake_fill	isolated areas
Administrative Boundaries	boundaries - hierarchy
Relief	hypsometric layers

Messages

Press space bar to continue



map plotting finished
Press F10 to continue

Figure 8.8 q & r

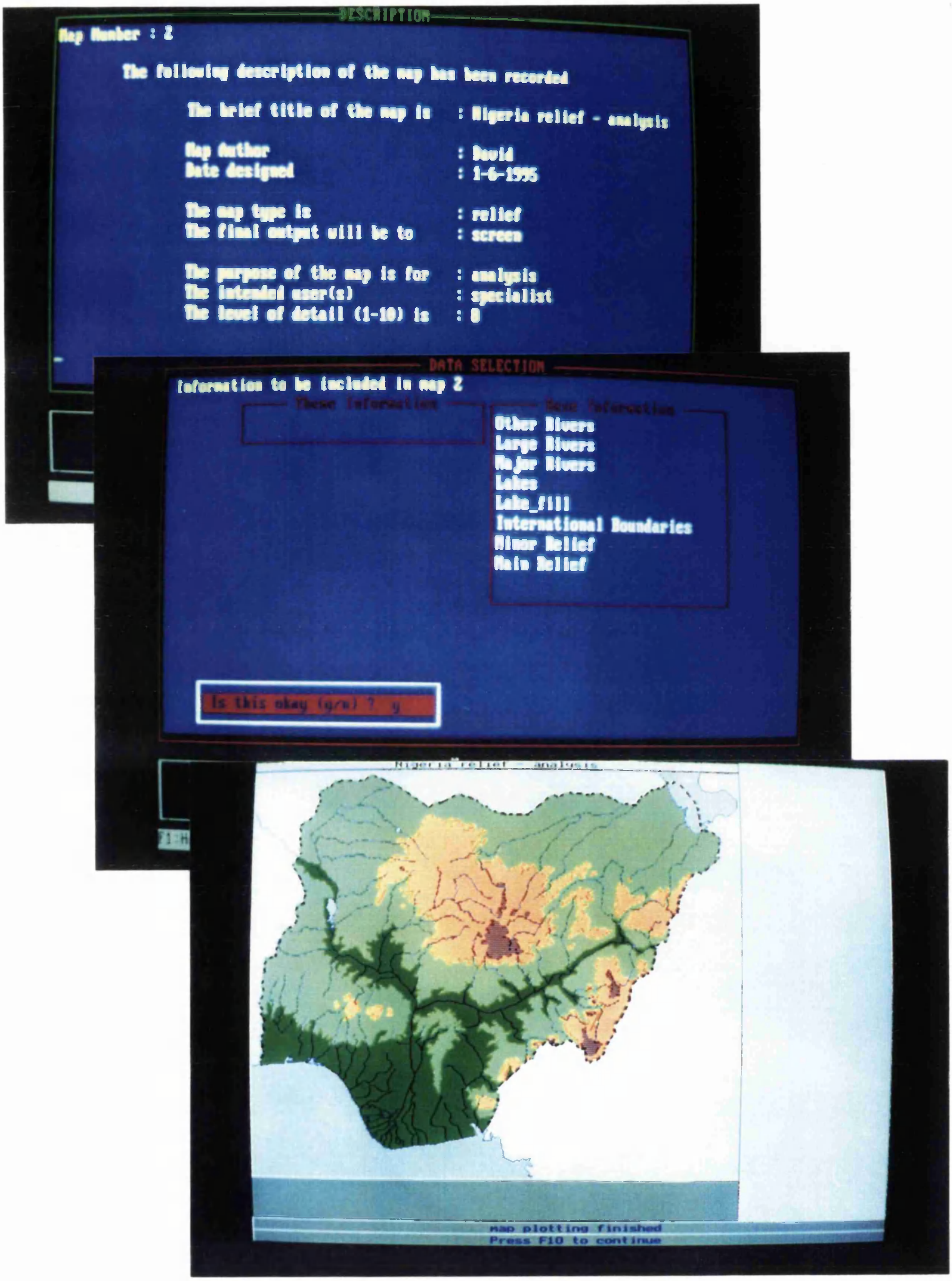
The representation screen and a relief map of Nigeria

Figure 8.9

The map frame for a relief map of Nigeria

```
fmap_date(dated(1995,6,15))
fmap_author("df")
fmap_title("Nigeria Relief Overview")
fmap_type("relief")
fmap_purpose("overview")
fmap_user("author")
foutput_media("screen")
flevel_of_detail(4)
fscale(7500000)
fformat(18,16.5)
flat_long(2.5,15,3.5,14)
flimits(277.98731656,388.9402453,1667.9238994,1541.2843966)
fselect_index(9)
fbase_info_list(["Coastline","Seas","Large Rivers","Major
          Rivers","Lakes","Lake_fill","International Boundaries","Main Relief"])
fbase_info("Main Relief",10)
fbase_info("International Boundaries",10)
fbase_info("Lakes",10)
fbase_info("Large Rivers",9)
fbase_info("Major Rivers",10)
fbase_info("Coastline",10)
ftheme_info_list([])
frepresentation("Coastline","boundaries - one level")
frepresentation("Seas","isolated areas")
frepresentation("Rivers","network - branching")
frepresentation("Lakes","boundaries - one level")
frepresentation("Lake_fill","isolated areas")
frepresentation("Administrative Boundaries","boundaries - hierarchy")
frepresentation("Relief","hypsometric layers")
fsymbolism("Coastline","Coastline",
          symbolspec("line","blue","dark","low","continuous","",0,"fine",201))
fsymbolism("Seas","Seas",symbolspec("area","cyan","light","low","solid","",0,"",158))
fsymbolism("Rivers","Major Rivers",
          symbolspec("line","blue","dark","mid","continuous","",0,"fine",190))
```

```
fsymbolism("Rivers","Large Rivers",
        symbolspec("line","blue","dark","mid","continuous","",0,"fine",190))
fsymbolism("Administrative Boundaries","International Boundaries",
        symbolspec("line","grey","dark","low","chain","chain",0,"thick",210))
fsymbolism("Lakes","Lakes",
        symbolspec("line","blue","dark","mid","continuous","",0,"fine",202))
fsymbolism("Lake_fill","Lake_fill",
        symbolspec("area","cyan","light","low","solid","",0,"",191))
fsymbolism("Relief","0-500",
        symbolspec("area","green","mid","low","solid","",0,"",141))
fsymbolism("Relief","500-2000",
        symbolspec("area","brown","mid","low","solid","",0,"",141))
fsymbolism("Relief","over 2000",
        symbolspec("area","brown","mid","mid","solid","",0,"",141))
```



```
DESCRIPTION
Map Number : 2

      The following description of the map has been recorded

            The brief title of the map is    : Nigeria relief - analysis

            Map Author                       : David
            Date designed                    : 1-6-1995

            The map type is                  : relief
            The final output will be to      : screen

            The purpose of the map is for    : analysis
            The intended user(s)             : specialist
            The level of detail (1-10) is    : 8
```

```
DATA SELECTION
Information to be included in map 2
       These Information            Base Information
                                    Other Rivers
                                    Large Rivers
                                    Major Rivers
                                    Lakes
                                    Lake_fill
                                    International Boundaries
                                    Minor Relief
                                    Main Relief

  Is this okay (y/n) ?  y
```

```
Nigeria relief - analysis

map plotting finished
Press F10 to continue
```

Figure 8.9 a, b & c

The show_description and show_selection screens and a relief map of Nigeria

CHAPTER NINE
Colouring the Political Map

To give an example of the advantages of Prolog compared to procedural languages for solving cartographic problems, a classic situation in cartography is the colouring of a political map where each country must have a different colour from it neighbours. It can be proved that four colours are required, although frequently 5 or 6 are used. A solution for incorporating this type of map into the system would be to store a feature code for each country which would relate to a look-up table of colours, much as would be done for a geological map or a land use map. However, the coding used would be arbitrary and not responsive to changes in the number of colours used, the level of hierarchy to show, etc. Manually assigning such codes would also be time consuming and is not true automation of the problem.

The general description of how to solve this problem, assuming four colours, is as follows:

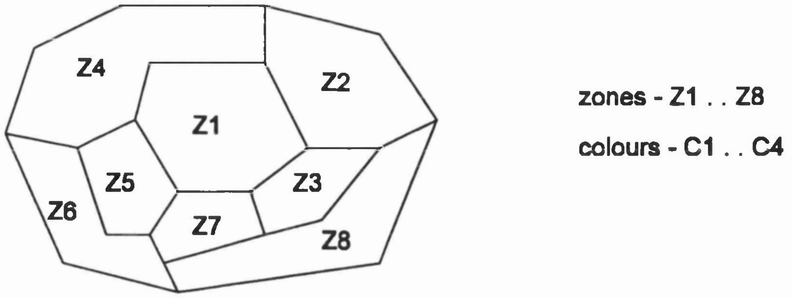
```
Assign colour 1 to zone 1;
For zone x  (x = 2 .. n)
    assign colour 1 to zone x;
    if adjacent to any zone previously assigned the same colour, try the next
    colour;
    if no clashes proceed to zone x+1 else
        if no more colours available for zone x, go back and try next
        colour for zone x-1
```

The sequence of processing and the resulting map is illustrated in Figure 9.1.

In Prolog, with a slight restructuring of the problem, the solution is trivial for a specific example case. All that is required is two predicates, one which defines two zones as being adjacent and one which assigns colours to a list of zones:

```
next(zone1,zone2)
colour([zone_list])
```

Using the first of these predicates a list of the pairs of possible adjacent colours can be asserted as facts in working memory (recall that the use of the initial lower case letter for an object denotes a constant):



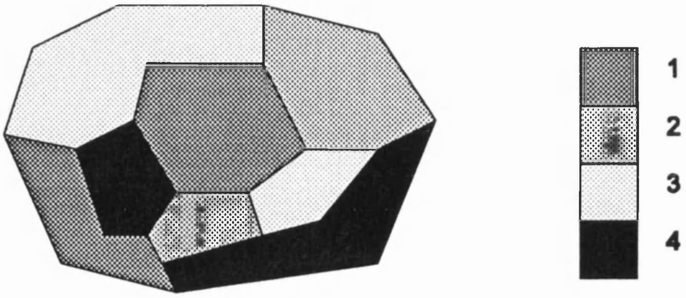
(a) Map of zones

	iteration 1		iteration 2		iteration 3
Z1	1				
Z2	~~1~~ 2				
Z3	~~1~~ ~~2~~ 3				
Z4	~~1~~ ~~2~~ 3				
Z5	~~1~~ 2				~~3~~ 4
Z6	~~1~~		~~2~~ ~~3~~ 4		1
Z7	~~1~~ ~~2~~ ~~3~~ 4	↑	~~1~~ ~~2~~ ~~3~~ ~~4~~	↑	~~1~~ 2
Z8	~~1~~ ~~2~~ ~~3~~ ~~4~~	↑			~~1~~ ~~2~~ ~~3~~ 4

(b) colour search process

/ conflict with adjacent zone, select next colour

↑ backtrack to previous zone



(c) final coloured map

Figure 9.1
Automated colour assignement for politcal map

```
            next(colour1, colour2)
            next(colour1, colour3)
            next(colour1, colour4)
            next(colour2, colour3)
                 etc.
```

In order to colour the map we need only define which regions are adjacent. The colours are assigned by successively trying to match each region with all those further down the list of regions to which it is adjacent with an allowable pair of colours. In this simplified example the eight zones of Figure 9.1a are used and it is assumed that the adjacencies are known:

```
       colour(Z1,Z2,Z3,Z4,Z5,Z6,Z7,Z8):-
                                    next(Z1,Z2) AND next(Z1,Z3) AND
                                    next(Z1,Z4) AND next(Z1,Z5) AND
                                    next(Z1,Z7) AND
                                    next(Z2,Z3) AND next(Z2,Z4) AND
                                    next(Z2,Z8) AND
                                    next(Z3,Z7) AND next(Z3,Z8) AND
                                    next(Z4,Z5) AND next(Z4,Z6) AND
                                    next(Z5,Z6) AND next(Z5,Z7) AND
                                    next(Z6,Z7) AND next(Z6,Z8) AND
                                    next(Z7,Z8).
```

Prolog's built in unification and backtracking mechanisms will process this list until a solution is found. This uses a depth first blind search procedure to find the solution. All permutations of next(Zx, Zy) included in the rule will be matched against the colour possibilities until it is possible to assign a valid result for each of the variables Z1 .. Z8.

There are some limitations to the above approaches, the main one being that colour one (or the first pair) is always tried first, then colour two, etc. The likely result of this is that there will be most zones with the first colour on the list and fewest of the last colour. Ideally there should be approximately the same number of zones with each colour. One solution to this problem is to arbitrarily assign each zone an initial colour with approximately the same number of zones being given each of the colours. The list of zones is then processed to check for conflicts of colour between adjacent zones. If a conflict is found, the colour of one of the zones is changed to the next colour on the list, and so on. An alternative approach is to create a list of the colours allowed and to select the next colour on the list for each new zone to be symbolised. This is achieved by rotating the list, i.e. once a colour has been used or tried for a zone it is moved to the end of the list. This makes the list of colours appear to be infinitely long. A check must be made to stop attempting to assign a colour to the current zone once all colours have been tried and to cause the system to backtrack to previous zones. This second method has been adopted.

Further optimisation could take place after all zones have been assigned a colour. A check could be made on the number of zones in each colour. If there was a large imbalance, each zone with the dominant colour could be checked to see if it could be assigned the least used colour, and so on.

In practice it is not a trivial matter to extend the above example to the general case where any number of regions may be required to be coloured using any number of colours, although using the list processing and backtracking capabilities of Prolog makes this easier than with procedural languages. The solution presented here assumes that the data for the zones is stored in a topologically structured file and that each of the chains (lines between zones) is coded with the left and right hand polygons, a fairly common feature of structured data in GIS and mapping systems such as GIMMS. From the chain file the set of adjacencies can be asserted into Prolog's working database. Also, rather than creating a list of adjacent colours that are allowed, the approach adopted allows any pair of colours to be adjacent, as long as the two colours are different.

In the listing that follows some of the supporting predicates are not fully listed, but their names indicate the function they perform.

Rather than assigning actual colours, the procedure here is to assign each zone to a class or group. These classes are then matched to the colours set for each group in the symbolism module, i.e. in symbolism the only concern was how many groups (colours) there were to be and their graphic variables, not which zones were assigned which class or colour. Having assigned each zone to a colour group, the plotting is carried out in the same way as categorical or choropleth maps.

```
/*    POLIT4.PRO        5/10/94                        */
/*    modularised version of polit.pro                 */
DATABASE - LOCAL
```

```
   wcolour(             % relates zone name to group
            SYMBOL,           % zone
            SYMBOL)           % class (colour)
   wadjacent(           % database term for adjacent zones
            SYMBOL,           % left zone
            SYMBOL)           % right zone
   wclasses(            % groups to be created
            INTEGER,          % number of classes
            SYMBOLLIST)       % class (colour) names
PREDICATES
   sort_adjacent(       % the main predicate
            SYMBOL)           % feature class to be coloured
   get_adjacent(        % gets adjacent zone pairs from
                        % database and asserts locally
            SYMBOL)           % feature class
   repeatread(          % creates file reading loop
                  FILE)             % name of coordinate file
   get_zones(                 % compile list of zones to colour
            SYMBOLLIST)
   writecolours         % saves colour for each zone

   sort_adjacent_classes2(  % main recursive clause -
                                    % works thru' list of zones
            SYMBOLLIST)       % list of zone names
   nextcolour(          % rotates list of colours and
                              % assigns next colour to current zone
            SYMBOL)           % current colour
   no_clash(                  % checks for conflict between zones
            SYMBOL,           % zone 1
            SYMBOL)           % zone 2

CLAUSES     % note: bold used to indicate the start of new clause
sort_adjacent(F_Class):-      % main clause - overall control
   get_adjacent(F_class),           % get data from database
   get_zones(Zones),                % make list of zones
   findall(Class,fsymbolism(F_class,Class,_),Classes),
                                    % make list of class names
   length_of(Classes,Num),          % get number of classes
   assert(wclasses(Num,Classes)),
   !,                          % cut - don't backtrack beyond here
   sort_adjacent_classes2(Zones),   % call predicate that does
                                    % the work
   !,
   writecolours,                    % save the class for each zone
   retractall(_,local).             % clear the working database

sort_adjacent_classes2([]).         % empty list -
                                    % all zones assigned
sort_adjacent_classes2([Z1|Zones]):-  % main working clause
                                    % works on zone Z1
   retractall(wcolour(start,_)),    % clear colours
   wclasses(_,[Start|_]),           % get first colour
   asserta(wcolour(start,Start)),
                              % save this as starting colour for
                              % this zone & don't cycle past it
   nextcolour(Col1),          % rotate list of colours to bring next
                              % colour to head of list for use if
                              % backtracks - if no further colours
                              % go back to previous zone
   retractall(wcolour(Z1,_)),       % delete any colours already
                              % set for this zone - essential
                              % if backtracked beyond this zone
                              % has no effect on first pass
   no_clash(Z1,Col1),         % check all zones adjacent to
                              % this one for conflict
                              % if fail, go back and try next colour
   assertz(wcolour(Z1,Col1)),       % no conflicts with this
                                    % zone so save colour for it
   sort_adjacent_classes2(Zones).   % process rest of zones

nextcolour(Colour):-                % first clause -
                                    % rotates list of colours
   wclasses(Num,[Colour|Colours]),  % get list of colours
   append(Colours,[Colour],New),    % move first to last
   retractall(wclasses(_,_)),       % delete old list
   asserta(wclasses(Num,New)).      % save new list
```

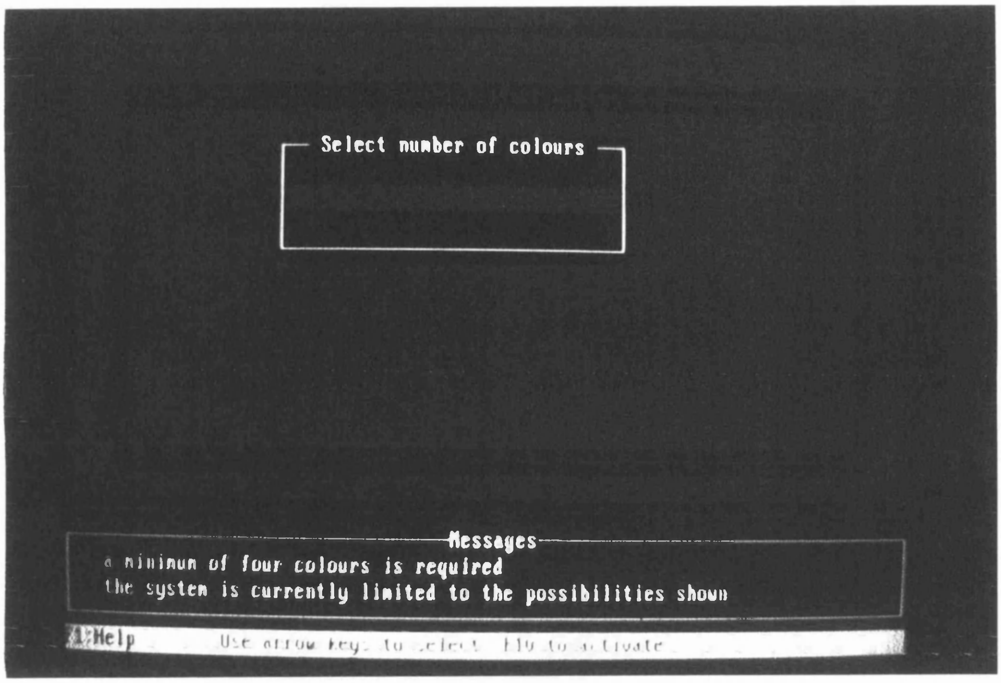
```
nextcolour(Colour):-                % called on backtrack
                                    % checks to see if back to start
                                    % colour. If so fails and forces
                                    % backtrack to previous zone
   wclasses(_,[Start|_]),           % get start colour
   not(wcolour(start,Start)),       % check not completed rotation
                                    % of list
   nextcolour(Colour).              % rotate list

no_clash(Zone,Colour):-       % check for forward clash
   wadjacent(Zone,Z),         % looks for adjacent zone
   wcolour(Z,Colour),         % succeeds if adjacent zone has same
                              % colour, else backtrack and look
                              % for other adjacencies
   !,                         % cut very important here. If passes
                              % this, then don't look for other
                              % clauses for this predicate
   fail.                      % if get here, predicate fails
no_clash(Zone,Colour):-       % as above but reverse order of zones
   wadjacent(Z,Zone),
   wcolour(Z,Colour),
   !,
   fail.
no_clash(_,_).                % succeed - no clashes found
                              % proceed to next zone

get_adjacent(_):-                   % get adjacency data
   kmeta_data(F_class,_,_,_,_,_,_,Coordname,_,_,_),
   kcoord_file(Coordname,_,_,_,_,_,_,_,_,Coordfile,_),
   file_exist(Coordfile),           % check to see file found
   openread(input,Coordfile),       % open file for reading
   readdevice(input),               % make file input device
   repeatread(input),         % creates loop on backtrack until end
                              % of file is reached then fails
   readterm(plotdata,chain(_,Coordname,Right,Left,_,_,_,_)),
                                    % read next line
   assertz(wadjacent(Right,Left)),  % add adjacency to database
   fail.                            % force backtrack to next term
```

```
                                      % in coordinate file
get_adjacent(_):-               % finished reading file
   readdevice(keyboard),              % set input back to keybd
   closefile(input).                  % close coordinate file

repeatread(_).                  % first call - always succeeds
repeatread(File):-              % called on backtrack
                                % succeeds if not end of file
   not(eof(File)),              % checks for end of file
   !,
   repeatread(File).            % continues file reading loop

get_zones(Zones):-              % compiles unique list of zones
                                      % to be coloured
    findall(A,wadjacent(A,_),As),     % get all left hand zones
    findall(B,wadjacent(_,B),Bs),     % get all right hand zones
    append(As,Bs,Cs),                 % add above lists
    uniquelist(Cs,Zones).             % remove duplicates

writecolours:-
   openwrite(output,"diags.lis"),
   writedevice(output),
   wcolour(State,Colour),
   nl,write(State,"  ",Colour),
   fail,!.
writecolours:-
   writedevice(screen),
   closefile(output).
```

An example of the goal passed to this module would be:

```
sort_adjacent_classes("Administrative areas").
```

The menus associated with this map topic for selecting the number of colours and the basic colour scheme are shown in Figures 9.2 a & b. A map of Nigeria with only politically coloured zones and the sea symbolised is shown in Figure 9.3 and a map using default values for all slots is illustrated in Figure 9.4. The frame for this final map is given in Figure 9.5.

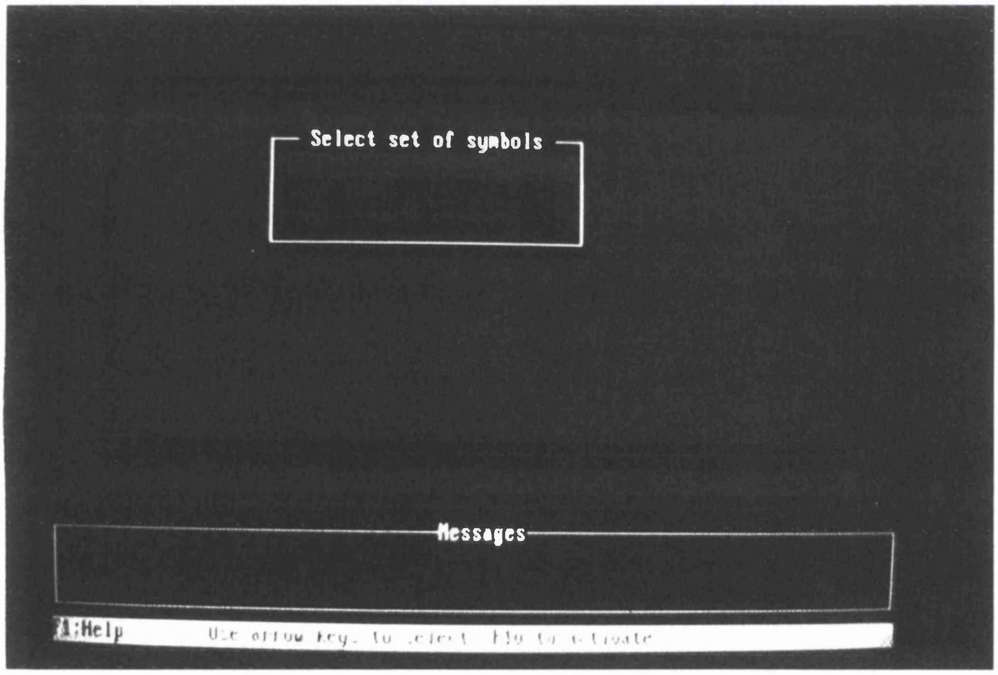


Select number of colours

Messages

a minimum of four colours is required
the system is currently limited to the possibilities shown

1:Help   Use arrow keys to select F10 to activate



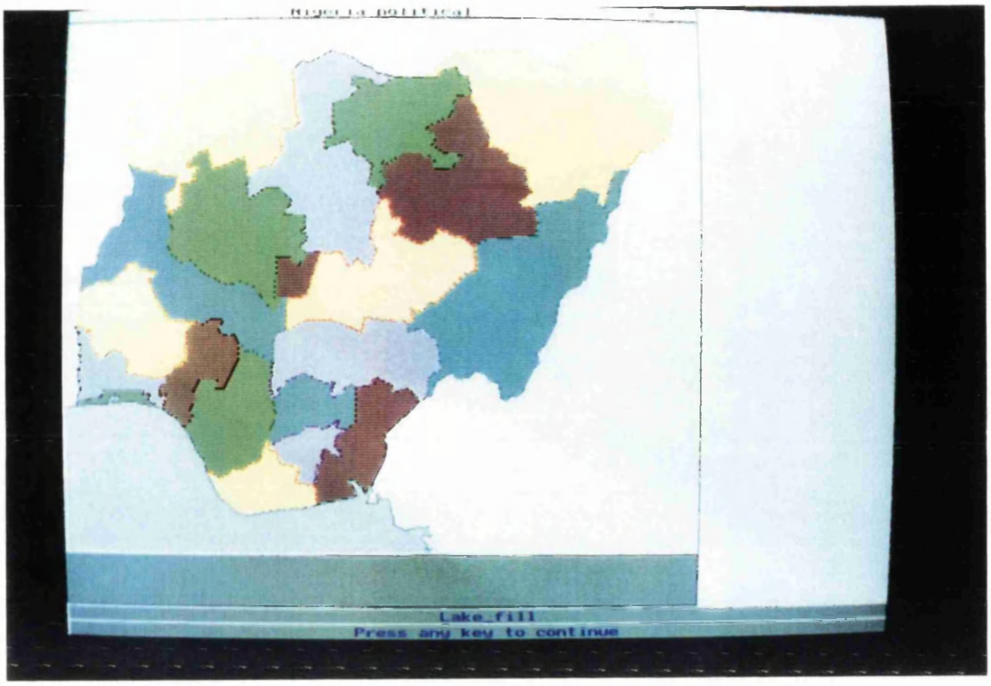
Select set of symbols

Messages

1:Help   Use arrow keys to select F10 to activate

Figure 9.2

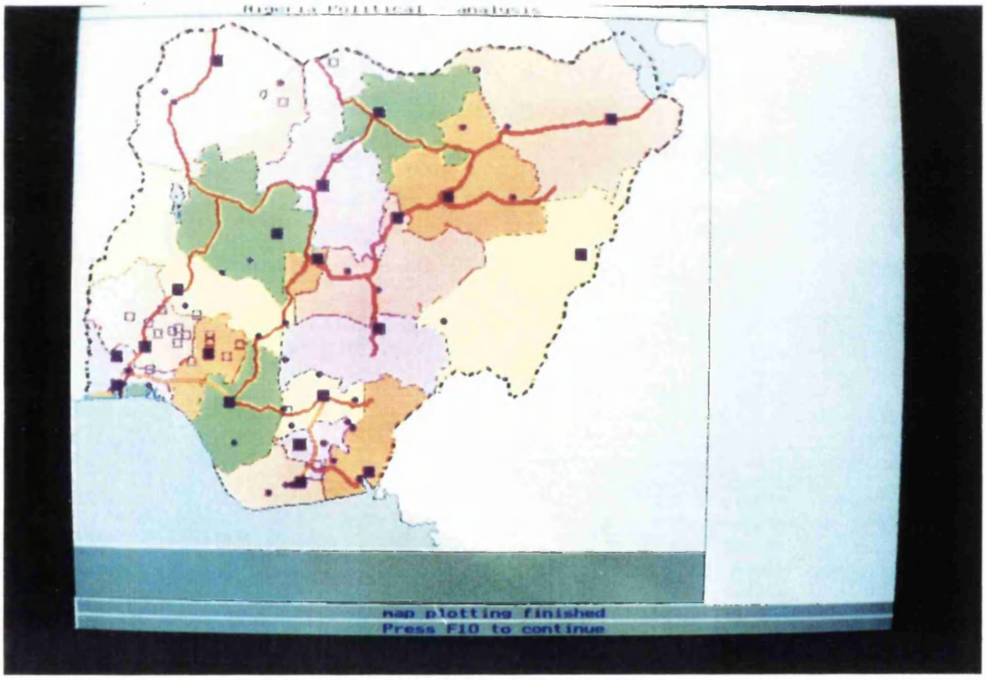
a) Number of colours selection screen; b) colour set selection screen



```
Lake_fill
Press any key to continue
```

Figure 9.3

An example of politically coloured zones



```
nap plotting finished
Press F10 to continue
```

Figure 9.3

A political map of Nigeria

Figure 9.5

The map frame for a political map of Nigeria.

```
fmap_date(dated(1995,6,15))
fmap_author("df")
fmap_title("Nigeria Political")
fmap_type("political")
fmap_purpose("analysis")
fmap_user("author")
foutput_media("screen")
flevel_of_detail(8)
fscale(7500000)
fformat(18,16.5)
flat_long(2.5,15,3.5,14)
flimits(277.98731656,388.9402453,1667.9238994,1541.2843966)
fselect_index(7)
fbase_info_list(["Coastline","Seas","Lakes","Lake_fill","State Boundaries",
        "International Boundaries","Minor Towns","Major Towns","Capitals",
        "Highways","Main Highways"])
fbase_info("Highways",7)
fbase_info("Main Highways",8)
fbase_info("Minor Towns",8)
fbase_info("Major Towns",9)
fbase_info("Capitals",10)
fbase_info("State Boundaries",9)
fbase_info("International Boundaries",10)
fbase_info("Lakes",9)
fbase_info("Coastline",10)
ftheme_info_list(["Administrative areas"])
frepresentation("Administrative areas","unclassed areas - hierarchy")
frepresentation("Coastline","boundaries - one level")
frepresentation("Seas","isolated areas")
frepresentation("Lakes","boundaries - one level")
frepresentation("Lake_fill","isolated areas")
frepresentation("Administrative Boundaries","boundaries - hierarchy")
frepresentation("Settlements","ranked points")
frepresentation("Roads","network - link & node")
```

```
fsymbolism("Coastline","Coastline",
        symbolspec("line","blue","dark","low","continuous","",0,"fine",201))
fsymbolism("Seas","Seas",symbolspec("area","cyan","light","low","solid","",0,"",158))
fsymbolism("Administrative Boundaries","International Boundaries",
        symbolspec("line","grey","dark","low","chain","chain",0,"thick",210))
fsymbolism("Administrative Boundaries","State Boundaries",
        symbolspec("line","black","dark","low","chain","chain",0,"fine",210))
fsymbolism("Administrative areas","group1",
        symbolspec("area","green","light","mid","solid","",0,"",102))
fsymbolism("Administrative areas","group2",
        symbolspec("area","yellow","pale","mid","solid","",0,"",102))
fsymbolism("Administrative areas","group3",
        symbolspec("area","orange","light","mid","solid","",0,"",102))
fsymbolism("Administrative areas","group4",
        symbolspec("area","red","light","mid","solid","",0,"",102))
fsymbolism("Administrative areas","group5",
        symbolspec("area","magenta","pale","mid","solid","",0,"",102))
fsymbolism("Lakes","Lakes",
        symbolspec("line","blue","dark","low","continuous","",0,"fine",202))
fsymbolism("Lake_fill","Lake_fill",
        symbolspec("area","cyan","light","low","solid","",0,"",191))
fsymbolism("Settlements","Capitals",symbolspec("point",
        "magenta","dark","mid","geometric","square",0,"medium",251))
fsymbolism("Settlements","Major Towns",
        symbolspec("point","magenta","dark","mid","geometric","box",0,"small",251))
fsymbolism("Settlements","Minor Towns", symbolspec("point",
        "magenta","dark","mid","geometric","dot",0,"v_small",251))
fsymbolism("Roads","Main Highways",
        symbolspec("line","red","mid","mid","cased","main_highways",0,"thick",223))
fsymbolism("Roads","Highways",
        symbolspec("line","red","mid","mid","cased","highways",0,"thick",223))
```

CHAPTER TEN

Summary and Conclusions

> The cartographic expert system would allow the user to specify scales, projections, colors, symbols and other map elements, but would make good decisions about defaults for any of these if the user chose not to specify them. Ideally such a system should ***even*** [italics mine] make an appropriate choice of the type of cartographic representation . . . Clearly, the development of such a cartographic expert system is a monumental task. In fact, it may be a problem which can be effectively addressed only by a multi-investigator team over a period of several to many years.[1]

The primary aims of this study have been to examine the application of Artificial Intelligence to cartographic design and to create a knowledge based system for producing small scale maps from a database, with minimum intervention from the system user. As can be seen from the examples in Chapters Eight and Nine and Appendix D, the system developed can produce a range of satisfactory maps of different topics at different scales. The user of the system only has to respond to a small number of simple questions, generally presented in the form of menus each of which gives sensible and context sensitive default values. No knowledge of cartographic design is required to produce sensible maps. The maps produced are responsive to the user's requirements with content and representation determined largely automatically from the simple user input and the rules built into the knowledge base. Thus, it can be concluded that the primary objective has been successfully achieved.

In order to achieve this goal, the map design process has been formally described, with particular emphasis given to the classification of cartographic representation methods that are commonly employed. Comprehensive rules for each stage in the design process and for each representation method have been developed and set out in the form of a functional specification for cartographic design expert systems. These rules have then been implemented in an interactive map design expert system using the Prolog language.

[1] Buttenfield & McMaster, 1991; 146

Trial runs of the system certainly did lead to some refinement of the knowledge base in order to match the author's expectation of what should have resulted, but these adjustments were not of a major nature, and some were to be expected. For example, some changes were made to the base information scores in the map_content fact for some map topics as a result of seeing maps on the screen. Generally, adjustments were to reduce the number of feature classes included in the maps as some screen displays were rather cluttered. Other adjustments to map_content reflected a refinement of the scores for more specific map topics lower down the hierarchy, rather than adopting the values for the higher 'class' of topic. Other similar adjustments occurred elsewhere in the knowledge base, but, in the author's opinion, overall a high level of success was achieved with much of the initial rule base.

As mentioned in Chapter Eight, the most problematic area is that of symbolisation and several attempts at implementing the rules for symbolisation developed in Chapter Five and Six were necessary to create an operational system. The basic structure outlined in these earlier chapters was not faulty, rather it was the translation from descriptive rules to a specific implementation that caused the difficulties. The rules as presented could be incorporated into other systems using different development environments.

Apart from these specific conclusions about the system, some broader observations can be made. From Chapter Three it can be seen that there has been a great deal of interest in the application of expert systems to cartography in the last ten years, but apart from a few selective, very narrow systems, few actual working cartographic expert systems have been described. Most of the literature is theoretical, discussing the potential (or pitfalls) of expert systems for cartographic problems, or how systems or knowledge might be structured. Many of the early writings were over-ambitious about what could easily be achieved and many show basic ignorance of the knowledge of cartographers and the practice of map design. Few have actually looked at the practice of cartography as it is carried out, but have based their suggestions on a restricted range of published literature and some vague concepts about producing maps. What the this study shows is that if one moves away from the general notion of designing any kind of map at any scale to placing some reasonable restrictions on what the system can be expected to achieve and by further refining the possibilities as the design progresses, then a solution is possible.

Apart from a few systems having very specialised functions, why are there still no commercial cartographic expert systems? Proper development of expert systems needs a great deal of time. Academics do not have enough time to devote to the detailed implementation of the theories, and software developers currently do not have (or at least do not see) the economic justification to devote the time. Monmonier reported in 1990 that Intergraph had invested heavily in developing expert systems for their mapping systems and GIS and had accumulated over 6,500 rules at that time.[2] Significantly however, despite this work and the large amount of published literature on expert systems for map generalisation, Intergraph's MapGeneralizer product introduced in 1994 does not use an expert system approach, but adopts Weibel's (1991) concept of 'amplified intelligence' where the system is seen as a toolbox using various rules with logical defaults built in, but does not operate in an inference mode. It is important also to note that the promotional brochure for MapGeneralizer stresses that it is designed for interactive use by operators *with a knowledge of cartographic generalisation.* This mirrors developments in Intelligent Computer Aided Design (ICAD) systems where the aim is to produce intelligent assistants rather than replace designers.

In discussing the problems of map design, Keates (1982) has stated that without adequate information to map we do not even have a start. One factor which has become increasingly evident as this project has progressed is the need for information *about* the information. It can equally be contended therefore, that without adequate metadata it is impossible to consider automating the process. Many of the decisions about representation faced by the cartographer are solved by his understanding of the world and the information he has about it. Clearly, being able to describe this information in clear, unambiguous terms is an essential basis for developing an automated process, and these processes could extend beyond map design. Thus, the formal structures for describing information set out in Chapter Five are considered to be the key to future development of automated map design systems. Although this knowledge has been captured 'manually' in this instance, the development of systems for automatically and/or interactively quantifying this information in a comprehensive manner is an obvious avenue to pursue in future research.

Buttenfield & Mackaness have stated that "Automating the graphic design process has received relatively little attention in computer cartography" (1991; 436).

[2] Monmonier, M.S. pers comm. 1990.

They view this lack of attention to map design as a growing contradiction given the sophistication of the equipment and processes involved in GIS. They go on to note that "Research has focused on those aspects of cartography that can be compartmentalized (symbolization, generalization and text placement)" (1991; 437) and believe that this isolation of elements from one another has reduced their usefulness in developing an 'integrated solution' to automated map design. The current study has concentrated on these broader issues and has solved many aspects of the design of the small scale maps under investigation. Clearly, however, the system developed is not yet a comprehensive mapping system. It does not deal with names; it does not generalise features, the only generalisation being at the simple level of feature selection; and it does not consider the overall layout of map components. Although not trivial, these elements can be considered as extensions to the core of a map design system. What the system does is tackle the basic composition of the map by determining the location, format and scale of the map, selecting appropriate classes of information and assigning representation methods to them. This is then followed by the specification of individual symbols. The approach of developing this core then extending it is more likely to succeed than developing the various components independently then trying to fit them together.

The database linked to the system has been developed specifically for the system, but it conforms to common practices of structured data suitable for GIS. One obvious development of the system described here would be to link it with commercial geographical databases, such as the Digital Chart of the World, or some of the increasing range of geo-referenced business databases becoming available on CD ROM. The other obvious route for development is to develop the system as a 'front end' to a GIS or mapping package. Many GIS do have a 'macro' language that allows for custom procedures to be added, but it is unlikely that these will allow the inference procedures built into the Prolog language to be incorporated, so a different approach to incorporating the knowledge would need to be taken. Thus, the research reported here could be considered to be just the beginning of a long term programme of development.

BIBLIOGRAPHY

Addis, T.R. (1985) **Designing Knowledge-Based Systems.** London: Kogan Page Limited.

Ahn, J., Freeman, H. (1983) A program for automatic name placement. *Proceedings, AutoCarto Six.* pp.426-434.

Akagi, S. (1991) Expert System for Engineering Design Based on Object-Oriented Knowledge Representation Concept. in Pham, D.T. (ed) **Artifical Intelligence in Design.** London: Springer-Verlag. pp.61-100.

Akervall, L., Degerstedt, K., Rystedt, B. (1991) Spatial metada systems at the National Land Survey of Sweden in: Medyckyj-Scott et al. (eds) **Metadata in the Geosciences.** pp.153-170.

Akman, V., ten Hagen, P., Rogier, J., Veerkamp, P. (1989) Knowledge Engineering in Design. In Gero, G.S. (ed) **AI Methods in Design.** Berlin: Springer Verlag pp.118-141.

Akman, V., ten Hagen, P.J.W., Tomiyama, T. (1990) A fundamental and theoretical framework for an intelligent CAD system. *Computer-Aided Design.* Vol.22, No.6. pp.352-367.

Aleksander, I. (ed.). (1985) **Advanced Digital Information Systems.** Englewood Cliffs: Prentice-Hall Inc.

Alty, J.L., Coombs, M.J. (1984) **Expert Systems: Concepts and Examples.** Manchester: The National Computer Centre Ltd.

Anthony, R., Emmerman, P.J. (1986) Spatial Reasoning and Knowledge Representation. In: Optiz, B. (ed.) **Geographic Information Systems in Government.** Hampton, Va.: A Deepak Publishing. pp.795-813.

Arbab, F. (1987) A Paradigm for Intelligent CAD. In: ten Hagen, P.J.W., Tomiyama, T. (eds.) **Intelligent CAD Systems I.** Berlin: Springer-Verlag. pp.20-39.

Armstrong, M.P. (1989) Interactive analytical displays for spatial decision support systems. *Proceedings, Autocarto 9.* pp.171-180.

Armstrong, M.P., Bennett, D.A. (1990) A knowledge based object-oriented approach to cartographic generalisation. *Proceedings GIS/LIS'90.* pp.48-57.

Bahill, A.T., Ferrell, W.R. (1986) Teaching an Introductory Course in Expert Systems. *IEEE Expert.* Vol.1, No.4. pp.59-63.

Barr, R. (1993) I can't see what you mean. *GIS Europe.* Vol.2,No.5. pp.16-17.

Basden, A. (1984) On the Application of Expert Systems. In Coombs, M.J. (ed.), **Developments in Expert Systems.** London: Academic Press Inc. (London) Ltd. pp.59-75.

Basoglu, U. (1982) A new approach to automated name placement. *Proceedings, AutoCarto 5.* pp.103-112.

Beard, K., Mackaness, W. (1991) Generalization Operations and Supporting Structures. *Proceedings, AutoCarto 10.* pp.29-45.

Beech, G. (ed.). (1986) **Interactive Learning on the IBM-PC.** Wilmslow: Sigma Press.

Begg, V. (1984) **Developing Expert CAD Systems.** London: Kogan Page Ltd.

Berry, D.C., Broadbent, D.E. (1986) Expert Systems and the Man-Machine Interface. *Expert Systems.* Vol.3, No.4. pp.228-231.

Bertin, J. (1967) **Semiologie Graphique.** Paris: Gauthier-Villiers.

Bijl, A. (1987) Strategies for CAD. In: ten Hagen, P.J.W., Tomiyama, T. (eds.) **Intelligent CAD Systems I.** Berlin: Springer-Verlag. pp.2-19.

Borland International. (1986) **Turbo Prolog Owner's Handbook.** Scotts Valley, Ca.: Borland International, Inc.

Borland International. (1987) **Turbo Prolog Toolbox User's Guide and Reference Manual.** Scotts Valley, Ca.: Borland International, Inc.

Borland International. (1988) **Turbo Prolog Reference Manual V. 2.0.** Scotts Valley, Ca.: Borland International, Inc.

Born, G. (ed.) (1988) **Guidelines for Quality Assurance of Expert Systems.** London: Computer Services Association.

Bos, E.S. (1984) Systematic symbol design in cartographic education. *ITC Journal. part 1.* pp.20-28.

Bouille, F. (1984a) Architecture of a Geographic Structured Expert System. *Proceedings, International Symposium on Spatial Data Handling.* pp.520-543.

Bouille, F. (1984b) A Structured Expert System for Cartography Based Upon the HBDS. *Proceedings, Auto Carto Six,* Volume 2. pp.202-210.

Bouille, F. (1986) Interfacing Cartographic Knowledge Structures and Robotics. *Proceedings, Auto Carto London.* pp.563-571.

Bouille, F. (1988) Developing Strategies in GIS by problem-solving methods based on a structured expert system. *Proceedings, Eurocarto Seven.* pp.42-50.

Brachman, R.J., Levesque, H.J. (eds). (1985) **Readings in Knowledge Representation.** Los Altos: Morgan Kaufmann Publishers, Inc.

Bramer, M.A. (1982) A Survey and Critical Review of Expert Systems Research. In Michie, D. (ed). **Introductory Readings in Expert Systems.** New York: Gordon and Breach Science Publishers. pp.3-29.

Bramer, M.A. (1991) Artificial Intelligence, Knowledge Engineering and the rise of the Expert System. Professorial Inaugural Lecture, Portsmouth Polytechnic.

Bratko, I. (1982) Streamlining the Problem-Solving Processes. In Michie, D. (Ed.). **Introductory Readings in Expert Systems.** New York: Gordon and Breach, Science Publishers, Inc. pp.177-191.

Breuker, J., Wielinga, R. (1987) Use of Models in the Interpretation of Verbal Data. In Kidd, A.L. (ed.) **Knowledge Aquisition for Expert Systems.** New York: Plenum Press.

Broome, F.R. (1987) Automated Map Inset Determination. *Proceedings, AutoCarto 8.* pp.466-470.

Brown, A. (1993) Map design for screen displays. *Cartographic Journal.* Vol.30,No.2. pp.129-135.

Bundy, A., et al. (1983) **Alvey IKBS Research Theme: Intelligent Front End. Workshop Report No. 1.** London: The Alvey Directorate.

Bundy, A., Byrd, L., Mellish, C.S. (1985) Special-purpose, but domain independent, inference mechanisms. In Steels, L., Campbell, J.A. (eds). **Progress in Artificial Intelligence.** Chichester: Ellis Horwood Limited. pp.93-111.

Butler, R. (1988) The Use of Artificial Intelligence in GIS. *Mapping Awareness.* Vol.2, No.3. pp.33-38.

Buttenfield, B.P., Mackaness, W.A. (1991) Visualization. In: Maguire, D.J., Goodchild, M.F., Rhind, D.W. (eds) **Geographic Information Systems: Principles and Applications,** Vol.1. London: Longman. pp.427-443

Buttenfield, B.P., Mark, D.M. (1991) Expert Systems in Cartographic Design. in: Taylor, D.R.F. (ed) **Geographic Information Systems: the microcomputer and modern cartography.** Oxford: Pergamon Press. pp.129-150.

Buttenfield, B.P., McMaster, R.B. (eds) (1991) **Map Generalization: Making rules for knowledge representation.** Harlow: Longman.

Campbell, W.J., Goettsche, C. (1989) Development of an Intelligent Interface for Adding Spatial Objects to a Knowledge-Based GIS. *Godard Conference on Space Applications of Artificial Intelligence.* pp.239-247.

Cao, X., He, Z., Pan, Y. (1990) Automated design of house-floor layout with distributed planning. *Computer-Aided Design.* Vol.22, No.4. pp.213-222.

Central Computer and Telecommunications Agency. (1985) **Expert Systems: Some Guidelines.** London: H.M. Treasury.

Chandra, N., Goran, W. (1986) Steps Toward a Knowledge Basd Geographical Data Analysis System. In: Optiz, B. (ed.) **Geographic Information Systems in Government.** Hampton, Va.: A Deepak Publishing. pp.749-764.

Charniak, E., McDermott, D. (1985) **Introduction to Artificial Intelligence.** Reading, Mass: Addison-Wesley.

Chen, G. (1986) A Rule-Based Approach for Spatial Object Modelling and Task Management. *Proceedings, Auto Carto London.* pp.588-597.

Christ, F. (1975) Automatically Symbolized Output of Map Data Compiled and Selected form a Data Base. *NaKaVerm, Series II.* No.32. pp.5-15.

Christ, F. (1975) Fully Automated and Semi-Automated Interactive Generalization, Symbolization and Light Drawing of a Small Scale Topographic Map. *NaKaVerm, Series II.* No.33 pp.19-36.

Chubb, D.W.J. (1986) A Spatial Problem Solver and Its Associated Spatial Representation. In: Optiz, B. (ed.) **Geographic Information Systems in Government.** Hampton, Va.: A Deepak Publishing. pp.815-836.

Clark, D.W. (1987) Cadastral Overlay: Ontological Accuracy. *Proceedings, GIS 87.* pp.666-677

Clark, K.L. (1982) An Introduction to Logic Programming. In Michie, D. (ed). **Introductory Readings in Expert Systems.** New York: Gordon and Breach, Science Publishers, Inc. pp.93-112.

Clark, K.L., McCabe, F.G. (1984) **micro-PROLOG: Programming in Logic.** Englewood Cliffs: Prentice Hall International.

Clocksin, W.F., Mellish, C.S. (1981) **Programming in PROLOG.** Heidelberg: Springer-Verlag.

Coelho, H. (1985) The Paradigm of Logic Programming in a Civil Engineering Environment. *Computers and Artificial Intelligence.* Vol.4, No.2. pp.115-124

Coelho, H., Cotta, J., Pereira, L. (1980) **How to Solve it with PROLOG.** Lisbon: Ministerio Da Habitacao E Obras Publicas, Laboratorio Nacional De Engenharia Civil.

Conway, T., Wilson, M. (1988) Psychological Studies of Knowledge Representation. In: Ringland, G.A., Duce, D.A. (eds.) **Approaches to Knowledge Representation: An Introduction.** Letchworth, Herts.: Research Studies Press Ltd. pp.117-160.

Cook, A.C., Jones, C.B. (1990a) A Prolog Interface to a Cartographic Database for Name Placement. *Proceedings, Fourth International Symposium on Spatial Data Handling.* pp.701-710.

Cook, A.C., Jones, C.B. (1990b) A Prolog Rule-Based System for Cartographic Name Placement. *Computer Graphics Forum 9.* pp.109-126.

Cooly, R.E., Hobbs, M.H.W. (1992) An application of AI to computing class partition values for thematic maps. *Proceedings, 5th International Symposium on Spatial Data Handling.* pp.371-380.

Coombs, M.J. (ed.). (1984) **Developments in Expert Systems.** London: Academic Press.

Coombs, M.J., Alty, J. (1984) Expert Systems: an alternative paradigm. In: Coombs, M.J. (ed). **Developments in Expert Systems.** London: Academic Press. pp.135-157.

Couclelis, H. (1986) Artificial Intelligence in Geography: The Shape of Things to Come. *The Professional Geographer.* Vol.38, No.1. pp.1-11.

Cowen, D.J., Ehler, G.B. (1994) Incorporating multiple sources of knowledge in a spatial decision support system. *Proceedings, 6th International Symposium on Spatial Data Handling.* pp.60-72.

Coyne, R. (1990) Logic of design actions. *Knowledge-Based Systems.* Vol.3,No.4. pp.242-257.

Coyne, R.D., Rosenman, M.A., Radford, A.D., Balachandran, M., Gero, J.S. (1988) **Knowledge-Based Design Systems.** Reading Mass.: Addison-Wesley Publishing Company.

Crenhange, M., et al. (1984) EXPRIM: An Expert System to Aid in Progressive Retrieval from a Pictorial and Descriptive Database. In: **New Applications of Data Bases.** London: Academic Press. pp.43-61.

Cromp, R.F. (1990) Knowledge Aquisition Techniques for Spatial Reasoning about Satellite Imagery. *Proceedings, Fourth International Symposium on Spatial Data Handling.* pp.742-751.

Crosley, P. (1985) Creating User Friendly Geographic Information Systems Through User Friendly System Supports. *Proceedings, Auto Carto 7.* pp.133-140.

Crossland, M.D. (1090) Hydrologic - A prototype Geographic Information Expert System for Examining an Artificial Intelligent Application in a GIS Environment. *Proceedings GIS/LIS'90.* pp.225-233.

Cuädrado, C.Y., Cuadrado, J.L. (1985) Prolog Goes to Work. *Byte.* August. pp.151-158.

David, B.T. (1987) Multi-Expert Systems for CAD. In: ten Hagen, P.J.W., Tomiyama, T. (eds.) **Intelligent CAD Systems I.** Berlin: Springer-Verlag. pp.57-67.

Davis, E. (1986) **Representing and Acquiring Geographic Knowledge.** London: Pitman.

Davis, J.R., Nanninga, P.M. (1985) GEOMYCIN: Towards a Geographic Expert System for Resource Management. *Journal of Environmental Management.* Vol.21. pp.377-390.

Davis, R. (ed) (1986) **Intelligent Information Systems.** Chichester: Ellis Horwood Limited.

De Simone, M. (1986) Automated Structuring and Feature Recognition For Large Scale Digital Mapping. *Proceedings, Auto Carto London.* pp.86-95.

DeMers, M.N. (1986) A Knowledge Base Aquisition Strategy for Expert Geographic Information System Development. In: Optiz, B. (ed.) **Geographic Information Systems in Government.** Hampton, Va.: A Deepak Publishing. pp.837-850.

Dietterich, T.G., Ullman, D.G. (1987) FORLOG: A Logic-based Architecture for Design. In: Gero, J.S. (ed.) **Expert Systems in Computer-Aided Design.** Amsterdam: Elsevier Science Publishers. pp.1-17.

Dobson, M. (1984) Effective Color Display for Map Task performance in a Computer Environment. *Proceedings, International Symposium on Spatial Data Handling.* pp.332-348.

Dobson, M. (1984) Human Factors in the Cartographic Design of Real-Time Color Displays. *Proceedings, Auto Carto Six,* Volume 1. pp.421-426.

Dodd, T. (1987) Expert Prolog. *Systems International.* May. pp.51,54.

Doerschler, J.S., Freeman, H. (1989) An expert system for dense-map name placement. *Proceedings, AutoCarto 9.* pp.215-224.

Drinnan, C.H. (1986) Line Representation Requirements and Techniques for Geographic Information Systems. Opitz, B.K. (Ed.) **GIS in Government. Hampton.** Va.: A. Deepak Publishing. pp.935-944.

Drummond, J. (1988) Fuzzy sub-set theory applied to environmental planning GIS. *Proceedings, Eurocarto Seven.* pp.11-22.

Duhovnik, J. (1987) Systematic Design in Intelligent CAD Systems. In: ten Hagen, P.J.W., Tomiyama, T. (eds.) **Intelligent CAD Systems I.** Berlin: Springer-Verlag. pp.224-238.

du Sautoy, M. (1995) Do computers threaten the true spirit of mathematics. *The Times.* Monday, April 24th, p.16.

Eager, A. (1989) Expert-system shells. *PC Magazine* Vol.2(5). pp.54-73.

Earnshaw, R.A. (ed.) (1987) **Theoretical Foundations of Computer Graphics and CAD.** Berlin: Springer-Verlag.

Eastman, J.R. (1987) Graphic Syntax and Expert Systems *Technical Papers, ACSM-ASP Annual Convention.* pp.87-96.

Ebinger, L.R., Goulette, A.M. (1989) Automated names placement in a non-interactive environment. *Proceedings, AutoCarto 9.* pp.205-214.

Egenhofer, M.J., Frank, A.U. (1988) Designing Object-Oriented Query Languages for GIS: Human Interface Aspects. *Proceedings, Third International Symposium on Spatial Data Handling.* pp.79-96.

Elder, W.E. (1987) Structures as Models in Design and Development. In: Yoshikawa, H., Warman, E.A. (eds.) **Design Theory for CAD.** Amsterdam: Elsevier Science Publishers. pp.33-50.

Elmes, A.E., Cai, G. (1992) Data quality issues in user interface design for a knowledge-based decision support system. *Proceedings, 5th International Symposium on Spatial Data Handling.* pp.303-312.

Ennals, R. (1983) **Beginning micro-PROLOG.** Chichester: Ellis Horwood Ltd.

Essinger, R. (1985) Obtaining maps from geographic information systems - Issues in the design of the cartographic interface. *Unpublished Masters dissertation, SUNY, Buffalo.*

Essinger, R. (1986) The Philosophy and Requirements of Computer-Aided Graphic Design in Cartography. *Proceedings, Auto Carto London.* pp.189-196.

Estes, J.R., Sailer, C., Tinney, L.R. (1986) Applications of Artifficial Intelligence Techniques to Remote Sensing. *The Professional Geographer.* Vol.38(2) pp.133-141

Fairchild, D. (1987) The Display of Boundary Information: A Challenge in Map Design in an Automated Production System. *Proceedings, AutoCarto 8.* pp.456-465

Falcidiero, B., Gamboro, C. Singigaglia, P. (1983) Automatic colouring of maps according to the elevation. *Proceedings, AutoCarto Six.* pp.426-434.

Fazio, P., Bedard, C., Gowri, K. (1989) Knowledge-based system approach to building envelope design. *Computer-Aided Design.* Vol.21, No.8. pp.519-527.

Fenves, S.J., Baker, N.C. (1987) Spatial and Functional Representation Language for Structural Design. In: Gero, J.S. (ed.) **Expert Systems in Computer-Aided Design.** Amsterdam: Elsevier Science Publishers. pp.511-529.

Fikes, R., Kehler, T. (1985) The role of frame-based representation in reasoning. *Communications of the ACM.* Vol.28,No.9. pp.904-920.

Fisher, P.F., Mackaness, W.A. (1987) Are Cartographic Expert Systems Possible? *Proceedings, AutoCarto 8.* pp.530-534.

Fisher, P.F., Wilkinson, G.G. (1985) Towards the Knowledge-Based Integration and Presentation of Remotely Sensed Images and Geographic Data. *Proceedings, Advanced Technology for Monitoring and Processing Global Environmental Data.* pp.51-55.

Forrest, D. (1981) The design and perception of nonumerical point symbols. *Unpublished Masters thesis, Queen's University.*

Forrest, D. (1990) A Model of Cartographic Design for Expert System Applications. *Proceedings, Fourth International Symposium on Spatial Data Handling.* pp.752-761.

Forrest, D. (1991) Classifying Cartographic Representations for Cartographic Design Expert Systems. *Proceedings, ICA91.*

Forrest, D. (1992a) The development of a frame based cartographic design expert system. *Occasional Paper Series No.30.* Department of Geography & Topographic Science, University of Glasgow.

Forrest, D. (1992b) Map Generalization: making rules for knowledge representaion - a review. *Cartographic Journal.* Vol.29,no.2. pp.187-190.

Forrest, D. (1993) Expert systems and cartographic design. *Cartographic Journal.* Vol.30,No2. pp.142-148.

Forrest, D., Pearson, A.W. (1990) Information Sources in Map Design. In: Perkins & Parry (eds.) **Information Sources in Cartography.** London: Bowker-Saur. pp.168-188

Forrest, D., Pearson, A.W. (1994) Somewhere over the rainbow: map design and GIS. *Proceedings AGI94.*

Forsyth, R., Naylor, C. (1985) **The Hitch-Hiker's Guide to Artificial Intelligence.** London: Chapman and Hall/Methuen Ltd.

Frank, A.U. (1982) MAPQUERY: Data Base Query Language for Retrieval of Geometric Data and Their Graphical Representation. *Computer Graphics.* Vol. 16, No. 3. pp.199-207.

Franklin, R.L. (1986) The Exploitation of Digital Data Through Electronic Displays. *Proceedings, Auto Carto London,* Vol.2. pp.389-398.

Franklin, R.W., Wu, P.Y.F. (1987) A Polygon Overlay System in Prolog. Proceedings, *AutoCarto 8.* pp.97-106.

Freeman, H. (1991) Computer name placement. In: Maguire, D.J., Goodchild, M.F., Rhind, D.W. (eds) **Geographic Information Systems: Principles and Applications,** Vol.1. London: Longman pp.445-456

Freeman, H., Ahn, J. (1984) Autonap - An Expert System for Automatic Name Placement. *Proceedings, International Symposium on Spatial Data Handling.* pp.544-569.

Frost, R. (1986) **Introduction to Knowledge Based Systems.** London: William Collins Sons & Co. Ltd.

Gahegan, M.N., Roberts, S.A. (1988) An intelligent, object-oriented geographical information system. *International Journal of GIS.* Vol.2, No.2. pp.101-110.

Gairola, A. (1986) An Application of Expert Systems in Design. *Proceedings, 2nd International Expert Systems Conference.* pp.517-531.

Gardels, K.D. (1987) The Expert Geographic Knowledge System: Applying Logical Rules to Geographical Information. *Proceedings, AutoCarto 8.* pp.520-529.

Gero, J.S. (1985) Expert Systems in CAD. *Computer Aided Design.* Vol 17, No 9. pp.396-398.

Gero, J.S. (e.d.) (1987) **Expert Systems in Computer-Aided Design.** Amsterdam: Elsevier Science Publishers.

Gero, J.S. (e.d.) (1991) **Artificial Intelligence in Design.** Oxford: Butterworth-Heinemann Ltd.

Gero, J.S., Coyne, R.D. (1987) Knowledge-based Planning as a Design Paradigm. In: Yoshikawa, H., Warman, E.A. (eds.) **Design Theory for CAD.** Amsterdam: Elsevier Science Publishers. pp.339-373

Giaranto, J., Riley, G. (1989) **Expert Systems: Principles and Programming.** Boston: PWS-Kent Publishing Company.

Gibson, A.E. (1987) A Model Describing Options for Parallel Color/Data Structuring. *Technical Papers, ACSM-ASP Annual Convention.* pp.97-106.

Gill, G.A., Trigg, A.D. (1988) Enhancements to 'Canvas' to Enable Automatic Colour Series Selection for VDU Images. *Report No.10,* NUTIS, Reading.

Gilmartin, P. (1989) Tiling patterns for Chorpleth Maps on medium Resolution CRT's: An Empirical solution. *Proceedings, 13th ICA Conference.* pp.459-481.

Gilmartin, P. (1992) *Cartography and Geographic Information Systems.* Vol.19, No.1. pp.37-47.

Gilmartin, P., Shelton, E. (1989) Choropleth maps on high resolution CRTs: the effect of number of classes and hue on communication. *Cartographica.* Vol.26,No.2. pp.40-52.

Gittins, D. (1986) **Querry Language Systems.** London: Edward Arnold (Publishers) Limited.

Glaschnig, J. (1982) PROSPECTOR: an expert system for mineral exploration. In Michie, D. (ed.) **Introductory Readings in Expert Systems.** New York: Gordon and Breach.

Gooding, K., Forrest, D. (1990) An Examination of the Difference between the Interpretation of Screen Based and Printed Maps. *Cartographic Journal.* Vol. 27, No. 1. pp.15-19.

Goodwin, C.W. (1987) The On-Line Atlas: A GIS for Flight Simulation. *Proceedings, GIS 87 - 2nd International Conference on GIS.* pp.678-684.

Gould, J.D. (1968) Visual Factors in the Design of Computer-Controlled CRT Displays. *Human Factors.* Vol.10(4). pp.359-376.

Goulette, A.M. (1985) Cartographic Use of Dithered Patterns on 8-Color Computer Monitors. *Proceedings, Auto Carto 7.* pp.205-209.

Grabowski, H., Seiler, W. (1985) Techniques, Operations and Models for Functional and Preliminary Design Phases. In: Yoshikawa, (ed.) **Design and Synthesis.** Amsterdam: Elsevier Science Publishers B.V. pp.17-22.

Graklanoff, G.J. (1985) Expert System Technology Applied to Cartographic Processes. *Proceedings, ACSM/ASP Fall Technical Meeting.* pp.613-624

Green, D. (1993) Map output from geographic information and digital information processing sytems. *Cartographic Journal.* Vol.30,No.2. pp.91-96.

Grelot, J-P. (1986) Archaic Data Models or Hardware as Concept Killers. *Proceedings, Auto Carto London.* pp.572-577.

Groop, R., Smith, (1982) A dot matrix method of portraying mcontinuous statistical surfaces. *American Cartographer.* Vol.9,No.2.

Grossler, K. (1993) How to aquire reliable rules and knowledge for map design expert systems (mapdes)? *Proceedings, 16th International Cartographic Conference.* pp.1150-1159.

Guilfoyle, C. (1987) Inside the Engine. *Expert Systems User.* February. p.9.

Hammond, P. (1984) micro-PROLOG for Expert Systems. In: Clark and McCabe (eds.). **micro-PROLOG.** Englewood-Cliffs: Prentice-Hall International. pp.294-319.

Harmon, P., King, D. (1985) **Expert Systems.** New York: John Wiley and Sons, Inc.

Hayes-Roth, F., Waterman, D.A., Lenat, D.B. (1983) **Building Expert Systems.** Reading, Mass.: Addison-Wesley Publishing Company, Inc.

Heivly, C.G. (1986) Using Expert Systems Concepts to Fix USGS Digital Boundaries. *Proceedings, 2nd. International Symposium on Spatial Data Handling.* pp.572-582.

Hirsch, S.A. (1980) An Algorithm for Automatic Name Placement Around Point Data. *Unpublished Masters thesis,* State University of New York, Buffalo.

Hirsch, S.A. (1982) An Algorithm for Automatic Name Placement Around Point Data. *American Cartographer.* Vol.9,No.1. pp.5-19.

Hirsch, S.A., Glick, B.J. (1982) Design issues for an intelligent names processing system. *Proceedings, AutoCarto 5.* pp.337-346.

Hootsman, R.M., de Jong, W.M., van der Wel, F.J.M. (1992) Knowledge-supported generation of meta-information on handling crisp and fuzzy datasets. *Proceedings, 5th International Symposium on Spatial Data Handling.* pp.470-479.

Hopkins, L.D., Johnston, D.M. (1990) Locating Spatially Complex Activities with Symbolic Reasoning: An Object-oriented Approach. *Proceedings, Fourth International Symposium on Spatial Data Handling.* pp.762-771.

Horvath, M., Markus, A. (1985) Prototype of a Prolog-based Design Engine. In: Yoshikawa, (ed.) **Design and Synthesis.** Amsterdam: Elsevier Science Publishers B.V. pp.9-12.

Hsu, P., Beard, K. (1990) Deriving semantic knowledge of graphic objects through scale of measurement and symbol representation. *Proceedings, GIS/LIS'90.* pp.789-797.

Hsu, P. (1992) Spatial structure and design process: A step towards an ideal information system for spatial design and planning. *ACSM Technical Papers.* pp.123-132.

Hua, Y., Gao, J. (1993) The establishment of a thematic map design expert system PC-mappe. *Proceedings, 16th International Cartographic Conference.* pp.943-947.

Hudson, D. (1987) Knowledge Representation Using First Order Predicate Calculus. *Proceedings, ACSM/ASP Annual Convention.* pp.157-166.

Hutzler, E., Spiess, E. (1993) A knowledge-based thematic mapping system - the other way round. *Proceedings, 16th International Cartographic Conference.* pp.329-340.

ICL. (1984) **Adviser User Manual.** ICL.

Illert, A. (1988) Automatic recognition of texts and symbols in scanned maps. *Proceedings, Eurocarto Seven.* pp.32-41.

Irvine, T. (1985) Prolog in Perspective. *Systems International.* November. pp.88-89.

Ishikawa, H., et al. (1987) Designing A Knowledge-Based Natural Language Interface. *IEEE Expert.* Vol.2, No.2. pp.57-71.

Iudica, N.R. (1989) Expert Systems for Design: Basic Techniques. in Gero, G.S. (ed) **AI Methods in Design.** Heidelberg: Springer Verlag. pp.107-117.

Jaakkola, O., Sarjakoski, T., Blom, T., Laurema, M. (1990) From Satellite Data to Thematic Representation - A Knowledge-Based System for Cartographic Visualisation. *Proceedings, Fourth International Symposium on Spatial Data Handling.* pp.711-722.

Jackson, P., Lefrere, P. (1984) On the application of rule-based techniques to the design of advice-giving systems. In: Coombs (ed). **Developments in Expert Systems.** London: Academic Press. pp.177-200.

James, M. (1984) **Artificial Intelligence in Basic.** Sevenoakes, Kent: Butterworth and Co.(Publishers) Ltd.

Jankowski, P., Nyerges, T.L. (1989) Design Considerations for MaPKBS - Map Projection Knowledge-Based System. *American Cartographer.* Vol.16, No.1. pp.85-89.

Jansen, J.J., Puttgen, H.B. (1987) ASDEP: An Expert System for Electric Power Plant Design. *IEEE Expert.* Vol.2, No.1. pp.56-65.

Jocob, R.J.K. (1983) Using Formal Specifications in the Design of a Human-Computer Interface. *Communications of the ACM.* Vol.26, No.4. pp.259-264.

Johnson, D.S., Basoglu, U. (1989) The use of Artificial Intelligence in the automated placement of cartographic names. *Proceedings, AutoCarto 9.* pp.225-230.

Jones, C.B., Cook, A.C., McBride, J.E. (1991) Rule-based control of automated name placement. *Proceedings, 15th ICA Conference.* pp.675-679.

Jong, W.M. de, Wel, F.J.M. van der, (1990) Embedded Artificial Intelligence and Spatial Data Handling, Some Presearch and Prototyping Experiences. *Proceedings, Fourth International Symposium on Spatial Data Handling.* pp.723-731.

Kadmon, N. (1972) Automated Selection of Settlements in Map Generalisation. *Cartographic Journal.* Vol.9,No.2. pp.93-98.

Kalay, Y.E., Swerdloff, L.M., Harfmann, A.C. (1987) A Knowledge-based Computable Model of Design. In: Gero, J.S. (ed.) **Expert Systems in Computer-Aided Design.** Amsterdam: Elsevier Science Publishers. pp.203-223.

Karimi, H.A. (1989) Geographic Knowledge Base Management System (GKBMS): The Future Challenge in Geomatics. *Proceedings, Canadian National Conference on GIS.* pp.704-709.

Karimi, H.A., et al. (1987) A Relational Database Model for an AVL System and an Expert System for Optimal Route Selection. *Proceedings, AutoCarto 8.* pp.584-593.

Keates, J.S. (1982) **Understanding Maps.** Harlow: Longman

Keates, J.S. (1989) **Cartographic Design and Production (2nd Edition).** Harlow: Longman.

Keates, J.S. (1989) Expert Systems and Cartographic Design. Unpublished paper, University of Glasgow.

Keates, J.S. (1989) Neural Networks and Visual Perception. Unpublished paper, University of Glasgow.

Kelsey, R. (1988) Expert Systems as a Preliminary Design Tool for Drinking Water Supply in Developing Countries. *Science, Technology and Development.* Vol.6, No.1. pp.13-20

Kidd, A.L. (ed.) (1987) **Knowledge Aquisition For Expert Systems.** New York: Plenum Press.

Kidner, D.B., Jones, C.B. (1994) A deductive object-oriented GIS for handling multiple representations. *Proceedings, 6th International Symposium on Spatial Data Hnadling.* pp.882-900

Kilpalainen, T., Sarjakoski, T. (1993) Knowledge-based methods and multiple representation as means of on-line generalization. *Proceedings, 16th International Cartographic Conference.* pp.211-220.

Kitchen, H. (1986) Expert-ease - Computing with expert knowledge. In: Beach, G. (ed.) **Interactive Learning on the IBM-PC.** Wilmslow: Sigma Press. pp.239-245.

Klahr, P., Waterman, D.A. (1986) Artificial Intelligence: A Rand Perspective. *The AI Magazine.* Vol.7, No.2. pp.55-64.

Kocabas, S. (1987) A Handle on Prolog. *.EXE Magazine.* April pp.18-25.

Koegel, J.F. (1987) A Theoretical Model for Intelligent CAD In: ten Hagen, P.J.W., Tomiyama, T. (eds.) **Intelligent CAD Systems I.** Berlin: Springer-Verlag. pp.206-223.

Kolodner, J.L. (1984) Towards an understanding of the role of experience in the evolution from novice to expert. In: Coombs M.J. (ed.) **Developments in Expert Systems.** London: Academic Press. pp.95-116

Kottenstein, T. (1990) Concept and Prototype of a Symbol Reference System for the Production of Thematic Maps. *Proceedings, Fourth International Symposium on Spatial Data Handling.* pp.772-781.

Kowalski, R.A. (1979) **Logic for Problem Solving.** New York: Elsevier.

Kowalski, R.A. (1985) Logic Programming. *Byte.* August. pp.161-177.

Kozai, K. (1987) An Educational Aid to Interpret Aerial Photographs for Coastal Landforms - An Expert System Approach. *Technical papers, ACSM-ASP Annual Convention.* pp.80-86.

Lambird, B.A., Lavine, D., Laveen, L.N. (1984) Distributed Architecture and Parallel Non-directional Search for Knowledge-Based Cartographic Feature Extraction Systems. In Coombs, M.J. (Ed.), **Developments in Expert Systems.** London: Academic Press. 221

Lambourne, J,J., Sutherland, F.R. (1990) A Vector and Raster Based Mapping Tool on Micro-Computers. *Cartographic Journal.* Vol.27, No.1. pp.20-23.

Langlotz, C.P., Shortcliff, E.H. (1984) Adapting a Consultation System to Critique User Plans. In Coombs, M.J. (ed.), **Developments in Expert Systems.** London: Academic Press Inc. (London) Ltd. pp.77-94

Langran, G.E., Poiker, T.K. (1986) Integration of Name Selection and Placement. *Proceedings, 2nd. International Symposium on Spatial Data Handling.* pp.50-64

Lansdown, R.J. (1987) Graphics, Design and Artificial Intelligence. Earnshaw, R.J. **Theoretical Foundations of Computer Graphics and CAD.** Berlin: Springer-Verlag. pp.1153-1174

Lanter, D.P., Surbey, C. (1994) Metadata analysis of GIS data processing: a cae study. *Proceedings, 6th International Symposium on Spatial Data Handling.* pp.314-324.

Laurema, M., Jaakkola, O., Sarjakoski, T., Schylberg, L. (1991) Formalization of cartographic knowledge using an expert system shell. *Reports of the Finnish Geodetic Institute*, Helsinki.

Lawson, B. (1988) **How Designers Think:** the Design Process Demystified. London: Butterworth Architecture.

Ledgard, H., Singer, A., Whiteside, J. (1979) **Directions in Human Factors for Interactive Systems.** Berlin: Springer-Verlag.

Lee, F., Robinson, G.J. (1993) Development of an automated generalisation system for large scale topographic maps. *Proceedings, GIS Research UK.* pp.36-45.

Levine, R.I., Drang, D.E., Edelson, B. (1986) **A Comprehensive Guide to AI and Expert Systems.** New York: McGraw-Hill Book Company.

Liang, Q., Shi, Z., Han, P. (1993) An experimental research on color automatically design. *Proceedings, 16th International Cartographic Conference.* pp.1227-1234.

Liardet, M. (1987a) Prolog Power. *Personal Computer World.* February pp.128-132.

Liardet, M. (1987b) Teach Yourself Prolog. *Personal Computer World.* March pp.158-162.

Liardet, M. (1987c) Teach Yourself Prolog. *Personal Computer World.* April pp.152-158.

Lillywhite, J. (1991) Identifying available spatial metadata: the problem. in: Medyckyj-Scott et al. (eds) **Metadata in the Geosciences.** pp.3-11.

Lu, X.Z. (1993) An object-oriented expert system for quantitative cartography. *Proceedings, 16th International Cartographic Conference.* pp.147-149

Lundberg, C.G. (1989) Knowledge Aquisition and Expertise Evaluation. *Professional Geographer.* Vol.41(3). pp.272-283.

MacCallum, K.J., Duffy, A., Green, S. (1987) An Intelligent Concept Design Assistant. In: Yoshikawa, H., Warman, E.A. (eds.) **Design Theory for CAD.** Amsterdam: Elsevier Science Publishers. pp.301-317.

MacEachren, A. (1994) **Some Truth With Maps: a primer on symbolization and design.** Washington: Association of American Geographers.

Macey, S.M., et al. (1987) Critiquing and Redrawing Maps: Techniques to Enhance Cartographic Knowledge. *Journal of Geography.* Vol. 87(5). pp.162-167.

Mackaness, W.A. (1986) Detection and Heuristic Resolution of Spatial Conflicts in Digital Map Datasets. *Unpublished manuscript,* Kingston Polytechnic, Surrey.

Mackaness, W.A. (1994) Issues in resolving visual spatial conflicts in automated map design. *Proceedings, 6th International Symposium on Spatial Data Handling.* pp.325-340.

Mackaness, W.A., Fisher, P.F. (1987) Automatic Recognition and Resolution of Spatial Conflicts in Cartographic Symbolisation. *Proceedings, AutoCarto 8.* pp.709-718.

Mackaness, W.A., Fisher, P.F., Wilkinson, G.G. (1985) The Design of a Cartographic Expert System. Final Report for the Natural Environment Research Council.

Mackaness, W.A., Fisher, P.F., Wilkinson, G.G. (1986) Towards a Cartographic Expert System. *Proceedings, Auto Carto London.* pp.578-587.

Mackaness, W.A., Scott, D.J. (1987) The Problems of Operationally Defining the Map Design Process for Cartographic Expert Systems. *Research Paper RR-87-06,* School of Geography, Kingston Polytecnic.

Mackinlay, J. (1986) Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics.* Vol.5, No.2. pp.110-141.

Maeda, Y., Takeeshige, A., Koguchi, T., Tomiyama, T., Yoshikawa, H. (1985) Frame Operating System for CAD. In: Yoshikawa, H. (ed.) **Design and Synthesis.** Amsterdam: Elsevier Science Publishers B.V. pp.13-16.

Maggio, R.C. (1987) The Role of Geographic Information Systems in the Expert System. *Proceedings, GIS 87 - 2nd Annual International Conference on GIS.* ASPRS/ACSM pp.685-692.

Maher, M.L., Zhao, F. (1987) Using Experience to Plan the Synthesis of New Designs. In: Gero, J.S. (ed.) **Expert Systems in Computer-Aided Design.** Amsterdam: Elsevier Science Publishers. pp.349-369.

Mainguenaud, Portier. (1990) GIS Querry Languages.

Mark, D.M. (1986) Knowledge-Based Approaches for Contour-to-Grid Interpolation. *Proceedings, 2nd. International Symposium on Spatial Data Handling.* pp.225-234.

Mark, D.M., Buttenfield, B.P. (1988) Design Criteria for a Cartographic Expert System. *Proceedings, 8th Internatioanl Workshop on Expert Systems and Their Application,* V.2. pp-413-425

Martinez, A.A. (1989) Automated insetting: An expert component embedded into the Census Bureau's map production system. *Proceedings, AutoCarto 9.* pp.181-190.

Mason, D.C. et. al. (1986) Knowledge-Based Segmentation of Remotely-Sensed Images. *Proceedings, ISPRS Commision IV.* pp.501-510.

Matsyama, T. (1987) A Knowledge-Based Aerial Image Understanding System and Expert System for Image Understanding. *IEEE Transactions on Geoscience and Remote Sensing.* Vol.25, No.3 pp.305-315.

McCrary, S.W. et al. (1090) EXGIS - An Expert System for GIS Selection in Small Communities. *Proceedings GIS/LIS'90.* pp.562-572.

McGranahan, M. (1985) Symbol and Background Value Effects in Choropleth Maps for Color CRT Display. *Proceedings ACSM-ASP Fall Technical Meeting.* pp.356-363.

McGranahan, M. (1986a) Interaction of Saturation and Value in Color Choropleth Maps. *Technical Papers, ACSM-ASPRS Annual Convention.* pp.76-84.

McGranahan, M. (1986b) Color Selection to Ease Information Retrieval from CRT Maps. *Proceedings, 2nd International Symposium on Spatial Data Handling.* pp.451-458.

McKeown, D.M. (1986) Some Experiences Integrating Spatial Databases with Knowledge Based Systems. In: Optiz, B. (ed.) **Geographic Information Systems in Government.** Hampton, Va.: A Deepak Publishing. pp.765-774.

McKeown, D.M. (1987) The Role of Artificial Intelligence in the Integration of Remotely Sensed Data with Geographic Information Systems. *IEEE Transactions on Geoscience and Remote Sensing.* Vol.25, No.3. pp.330-348.

McMaster, R.B., Mark, D.M. (1991) The design of a graphical user interface for knowledge acquisition in cartographic generalization. *Proceedings GIS/LIS'91.* pp.311-320.

Menon, S., Smith, T.R. (1989) A Declarative Spatial Query Processor for Geographic Information Systems. *Photogrammetric Engineering and Remote Sensing.* Vol.55, No.11. pp.1593-1600.

Merrit, D. (1989) **Building Expert Systems in Prolog.** New York: Springer-Verlag.

Merry, M.J., Hammond, P. (eds.) (1984) **Alvey IKBS Research Theme: Expert Systems. Workshop Report No. 1**. London: The Alvey Directorate.

Michalski, R.S., Davis, J.H., Bisht, V.S., Sinclair, J.B. (1985) Plant/ds: an expert consulting system. In: Steels, L., Campbell, J.A. (eds). **Progress in Artificial Intelligence.** Chichester: Ellis Horwood Limited. pp.257-270.

Michie, D. (ed.). (1979) **Expert Systems in the Micro Electronic Age.** Edinburgh: Edinburgh University Press.

Michie, D. (ed.). (1982) **Introductory Readings in Expert Systems.** New York: Gordon and Breach.

Middelkoop, H., Miltenburg, J., Mulder, N.J. (1989) Knowledge engineering for image interpretation and classification: a trial run. *ITC Journal.* No.1. pp.27-32.

Miller, D., Morrice, J. (1991) An expert system and GIS approach to predicting changes in the upland vegetation of Scotland. *Proceedings GIS/LIS'91.*

Minsky, M. (ed.). (1968) **Semantic Information Processing.** Cambridge, Mass.: The MIT Press.

Mittal, S., Araya, A. (1986) A Knowledge-Based Framework for Design. *Proceedings, AAAI 86.* pp.856-865

Monmonier, M.S. (1981) Automated techniques in support of planning for the National Atlas. *American Cartographer.* Vol.8,No.2. pp.161-172.

Monmonier, M.S. (1986) Towards a Practicable Model of Cartographic Generalisation. *Proceedings, Auto Carto London,* Vol.2. pp.257-266

Monmonier, M.S. (1990) Graphically Encoded Knowledge Bases for Expert-Guided Feature Generalization in Cartographic Display Systems. *International Journal of Expert Systems.* Vol.3,No.1. pp.65-71.

Monmonier, M.S. (1991) **How to lie with maps.** Chicago: University of Chicago Press.

Montreuil, B. (1990) Requirements for representation of domain knowledge in intelligent environments for layout design. *CAD.* Vol.22, No.2. pp.97-108.

Morris, A.J. (1985) Adroit: An Intelligent Aircraft Designer. *Proceedings, First International Expert Systems Conference.* pp.333-338.

Morrison, A. (1994) Description of spatial data types *Unpublished manuscript,* University of Glasgow .

Morrison, C., Forrest, D. (in press) A study of point symbol design for computer based large scale tourist mapping. *Cartographic Journal.* forthcoming.

Morrison, J.L. (1974) A Theoretical Framework for Cartographic Generalization with Emphasis on the Process of Symbolization. *International Yearbook of Cartography.* Vol.XIV. pp.115-127.

Morse, B.W. (1987) Expert Systems Interface to a Geographic Information System. *Proceedings, AutoCarto 8.* pp.535-541.

Moss, D.S. (1987) Intelligent Databases. *Byte.* Vol.12, No.1. pp.97-106.

Mower, J.E. (1986) Name Placement of Point Features Through Constraint Propagation. *Proceedings, 2nd. International Symposium on Spatial Data Handling.* pp.65-73.

Muller, J-C. (1989) Challenges ahead for the mapping profession. *Proceedings, Autocarto 9.* pp.675-683.

Muller, J-C. (1990) Rule-Based Generalization: Potentials and Impediments. *Proceedings, Fourth International Symposium on Spatial Data Handling.* pp.317-334.

Muller, J-C. (1991) Generalization of spatial databases In: Maguire, D.J., Goodchild, M.F., Rhind, D.W. (eds) **Geographic Information Systems: Principles and Applications,** Vol.1. London: Longman pp.457-475

Muller, J-C., Johnson, R.D., Vanzella, L.R. (1986) A Knowledge-Based Approach for Developing Cartographic Expertise. *Proceedings, 2nd. International Symposium on Spatial Data Handling.* pp.557-571.

Muller, J-C., Mouwes, P.J. (1990) Knowledge acquisition and representation for rule based map generalization. *Proceedings GIS/LIS'90.* pp.58-67.

Muller, J-C., Peng, W., Wang, Z. (1993) Procedural, logical and neural nets tools for map generalisation. *Proceedings, 16th International Cartographic Conference.* pp.181-191.

Muller, J-C., Wang, Z. (1990) A Knowledge Based System for Cartographic Symbol Design. *Cartographic Journal.* Vol. 27, No.1. pp.24-30.

Mulvenna, M.D., Hughes, J.G. (1993) Applying Knowledge-Based Techniques to an Object-oriented Geographical Information System. Paper presented at GIS Research UK.

Munakata, T. (1986) Procedurally Oriented Programming Techniques in Prolog. *IEEE Expert.* Vol.1, no.2. pp.41-47.

Navin chandra, D. (1992) Innovative design systems, where are we and where do we go from here? Part I: Design by exploration. *The Knowledge Engineering Review.* pp.345-362.

Naylor, C. (1983) **Build Your Own Expert System.** Bristol: Arrowsmith Ltd.

Naylor, C. (1988) Expert? You will be. *Your Computer.* Aug/Sept pp.60-62.

Naylor, C. (1989) How to talk turkey about expert systems. *Honywell Intellect.* Vol.1(1). pp.15-19.

Nee, A.Y.C., Poo, A.N. (1991) Expert CAD Systems for Jigs and Fixtures. in Pham, D.T. (ed) **Artifical Intelligence in Design.** London: Springer-Verlag. pp.343-370.

Newton, S. (1986) A Role for Expert Systems in Building Design. *Proceedings, 2nd International Expert Systems Conference.* pp.175-182

Nickerson, E.G., Freeman, H. (1986) Development of a Rule-Based System for Automatic Map Generalization. *Proceedings, 2nd. International Symposium on Spatial Data Handling.* pp.537-556.

Nicolin, B., Gabler, R. (1987) A Knowledge Based System for the Analysis of Aerial Images. *IEEE Transactions on Geoscience and Remote Sensing.* Vol.23, No.3. pp.317-329.

Nilsson, N.J. (1971) **Problem Solving Methods in Artificial Intelligence.** New York: McGraw-Hill.

Noack, W., et al. (1986) Correlation of Synthetic Aperture Radar Data Using a Knowledge Based Processing System. *Proceedings, 2nd International Expert Systems Conference.* pp.221-227.

Norman, D.A. (1983) Design Rules Based on Analysis of Human Error. *Communications of the ACM.* Vol.26, No.4. pp.254-258.

Nyerges, T.L., Jankowski, P. (1989) A Knowledge Base for Map Projection Selection. *American Cartographer.* Vol.16, No.2. pp.29-38.

Nystued, J.D. (1984) Comment on "Artificial Intelligence and Its Applicability to Geographic Problem Solving." *Professional Geographer.* Vol. 36(3). pp.358-359.

O'Callaghan, J.F., Robertson, P.K. (1986) Colour Image Display of Geographic Data Sets Using Uniform Colour Spaces. *Proceedings, 2nd International Symposium on Spatial Data Handling.* pp.322-326.

O'Callaghan, J.F., Simons, L.W. (1984) Map Display Techniques for Interactive Colour Mapping. *Proceedings, International Symposium on Spatial Data Handling.* pp.316-319.

Ohsuga, S. (1989) Towards intelligent CAD systems. *Computer-Aided Design.* Vol.21, No.5. pp.315-337.

Optiz, B.K. (ed.) (1986) **Geographic Information Systems in Government.** Hampton, Va.: A Deepak Publishing.

Oxman, R., Gero, J.S. (1987) Using Expert Systems for Design Diagnosis and Design Synthesis. *Expert Systems.* Vol.4, No.1. pp.4-15.

Palmer, B. (1984) Symbolic Feature Analysis and Expert Systems. *Proceedings, International Symposium on Spatial Data Handling.* pp.465-478.

Papadias, D., Kavouras, M. (1994) Acquiring, representing and processing spatial relations. *Proceedings, 6th International Symposium on Spatial Data Handling.* pp.631-645.

Pavilin, C. (1988) Logic in Knowledge Representation. In: Ringland, G.A., Duce, D.A. (eds.) **Approaches to Knowledge Representation: An Introduction.** Letchworth, Herts.: Research Studies Press Ltd. pp.13-44.

Payne, E.C., McArthur, R.C. (1990) **Developing Expert Systems.** New York: John Wiley & Sons, Inc.

Peacegood, G., Wilkinson, G.G., Fisher, P.F. (1986) Knowledge Base Requirements for Automated Mapping from High Resolution Satellite Imagery. *Proceedings, ISPRS Commision IV.* pp.661-667.

Pearl, J. (1984) Heuristics. Reading, Mass.: Addison-Wesley Publishing Company.

Peuquet, D. (1984a) The Application of Artificial Intelligence Techniques to Very Large Databases. *Proceedings, Auto Carto Six,* Volume 1. pp.419-420.

Peuquet, D.J. (1984b) Data Structures for Knowledge-Based Geographic Information Systems. *Proceedings, International Symposium on Spatial Data Handling.* pp.372-391.

Peuquet, D.J. (1991) Methods for Structuring Digital Cartographic Data in a Personal Computer Environment. In Taylor, D.R.F. (ed). **Geographic Information Systems.** Oxford: Pergamon Press. pp.67-96.

Pfefferkorn, C. et. al. (1985) ACES: A Cartographic Expert System. *Proceedings, Auto Carto 7.* pp.399-407

Pham, D.T. (ed) (1991) **Artificial Intelligence in Design.** London: Springer-Verlag.

Pham, D.T., de Sam Lazaro, A. (1991) Knowledge-Based Design of Jigs and Fixtures. in Pham, D.T. (ed) **Artifical Intelligence in Design.** London: Springer-Verlag. pp.371-390.

Pham, D.T., Tacgin, E. (1991) Techniques for Intelligent Computer-Aided Design. in Pham, D.T. (ed) **Artifical Intelligence in Design.** London: Springer-Verlag. pp.5-27.

Pillinger, I., Hartley, P., Strugess, C.E.N., Dean, T.A. (1991) An Intelligent Knowledge-Based System for the Design of Forging Dies. in Pham, D.T. (ed) **Artifical Intelligence in Design.** London: Springer-Verlag. pp.319-342.

Poiker, T.K. (1981) A Knowledge Based System for the Study of Archival Information. Unpublished paper, Simon Fraser University.

Poiker, T.K., Squirrell, R., Xie, S. (1982) The Use of Computer Science and Artificial Intelligence in Cartographic Design. *Proceedings, Auto Carto V.*

Pountain, R. (1984) Prolog on Microcomputers. *Byte.* December. pp.355-362.

Prolog Development Center (1990) **PDC Prolog Toolbox, Version 3.20.** Copenhagen: Prolog Development Center.

Prolog Development Center (1992) **PDC Prolog Reference Guide, Version 3.3.** Copenhagen: Prolog Development Center.

Prolog Development Center (1992) **PDC Prolog User's Guide, Version 3.3.** Copenhagen: Prolog Development Center.

Pullar, D. (1987) Query Language for Spatial Relationships. *Proceedings, ACSM/ASP Annual Convention.* pp.180-192.

Quinlan, J.R. (1982) Semi-Autonomous Aquisition of Pattern-Based Knowledge. In Michie, D. (ed.), **Introductory Readings in Expert Systems.** New York: Gordon and Breach, Science Publishers, Inc. pp.192-207.

Rabbits, L., Wright, G. (1987) Card Tricks for Smart Players. *Expert Systems User.* Vol.2, No.1. pp.16-19.

Ranzinger, M. (1985) A Data Structure for a Geo-Expert System. *International Yearbook of Cartography.* Vol.XXV. pp.183-188.

Raper, J.F., Bundock, M.S. (1990) GIS user interfaces: A window on the future. *Proceedings, AGI90.*

Ratcliffe, W. (1988) A paradox, a paradox. *Amstrad Professional Computing.* November. pp.53-55.

Richardson, D.E. (1988) Database considerations for rule-based map feature selection. *ITC Journal.* No.2 pp.165-170.

Richardson, D.E. (1989) Rule based generalisation for base map production. *Proceedings, Canadian National Conference on GIS.* pp.718-739.

Ringland, G.A. (1988) Structured Object Representation - Schemata and Frames. In: Ringland, G.A., Duce, D.A. (eds.) **Approaches to Knowledge Representation: An Introduction.** Letchworth, Herts.: Research Studies Press Ltd. pp.81-100.

Ringland, G.A., Duce, D.A. (eds.) (1988) **Approaches to Knowledge Representation: An Introduction.** Letchworth, Herts.: Research Studies Press Ltd.

Rivers, R. (1986) Interactive Learning with Expert 4. In: Beach, G. (ed.) **Interactive Learning on the IBM-PC.** Wilmslow: Sigma Press. pp.245-250.

Roberston, P.K. (1988) Choosing Data Representations for the Effective Visualisation of Spatial Data. *Proceedings, Third International Symposium on Spatial Data Handling.* pp.243-252

Robinson, A.H., Sale, R.D. (1969) **Elements of Cartography** (Third edition). New York: John Wiley & sons Inc.

Robinson, A.H., Sale, R.D., Morrison, J. (1984) **Elements of Cartography** (Fourth edition). New York: John Wiley & sons Inc.

Robinson, G., Jackson, M. (1985) Expert Systems in Map Design. *Proceedings, Auto Carto 7.* pp.430-439.

Robinson, G.J., Zaltash, A. (1989) Applications of Expert Systems to Topographic Map Generalisation. *Proceedings, AGI89.*

Robinson, V.B. (1988) Expert Systems and Geographic Information Systems. *Alberta LRIS Newsletter.* Vol.9, No.1. pp.3-6.

Robinson, V.B., Frank, A.U. (1987a) Expert Systems Applied to Problems in Geographic Information Systems: Introduction, Review and Prospects. *Proceedings, AutoCarto 8.* pp.510-519.

Robinson, V.B., Frank, A.U. (1987b) Expert Systems for Geographic Information Systems. *Photogrammetric Engineering and Remote Sensing.* Vol.53, No.10. pp.1435-1441.

Robinson, V.B., Frank, A.U., Blaze, M.A. (1986) Expert Systems and Geographic Information Systems: Review and Prospects. *Journal of Surveying Engineering.* Vol.112, No.2. pp.119-130.

Robinson, V.B., Frank, A.U., Hassan, A.K., (1987) Expert Systems for Geographic Information Systems. *Artificial Intelligence Applications in Natural Resource Management.* Vol.1, No.1. pp.47-57.

Robinson, V.B., Thongs, D., Blaze, M. (1985) Natural Language in Geographic Data Processing. *Proc. Advanced Technology for Moitoring and Processing Global Environmental Data. Remote Sensing Society/ CERMA.* pp.67-73.

Robinson, V.B., Thongs, D., Frank, A.U., Blaze, M. (1986) Expert Systems and Geographic Information Systems: Critical Review and Research Needs. In: Optiz, B.K. (ed.) **Geographic Information Systems in Government.** Hampton, Va.: A. Deepak Publishing. pp.

Rosenman, M.A., Coyne, R.D., Gero, J.S. (1987) Expert Systems for Design Applications. In: Quinlan, J.R. (ed.) **Applications of Expert Systems.** Sydney: Turing Institute Press / Addison-Wesley Publishing Co. pp.66-84.

Roubal, J., Poiker, T.K. (1986) Automated Contour Labelling and the Contour Tree *Unpublished paper,* Simon Fraser University.

Ruzak Masur, E., Castner, H.W. (1990) Horton's ordering scheme and the generalisation of river networks. *Cartographic Journal.* Vol.27,No.2. pp.104-112.

Sagalowicz, D. (1984) Development of an Expert System. *Expert Systems.* Vol.1/2. pp.137-141.

Samson, L., Poiker, T.K. (1985) Graphic Design with Color Using a Knowledge Base. *Paper presented at the 10th CCA Annual Conference.*

Scheepers, C.F. (1989) Vector-based computer graphics in automated map compilation. *Proceedings, AutoCarto 9.* pp.724-734.

Schildt, H. (1987) **Advanced Turbo Prolog.** Berkeley, Ca.: Osborne/McGraw-Hill.

Schor, M.I. (1986) Declarative Knowledge Programming: Better than Procedural? *IEEE Expert.* Vol.1, No.1. pp.36-43.

Senthil Kumar, A., Nee, A.Y.C., Prombanpong, S. (1992) Expert fixture-design system for an automated manufacturing environment. *Computer Aided Design.* Vol.24,No.6. pp.316-326.

Shneiderman, B. (ed.). (1978) **Databases: Improving Usability and Responsiveness.** New York: Academic Press.

Siekierska, E. (1984) Towards an Electonic Atlas. *Proceedings, Auto Carto Six,* Volume 2. pp.464-474.

Siekierska, E.M. (1989) Advantages of expert systems with examples of the use of the "VP.Expert" system for cartographic applications. *Proceedings, Canadian National Conference on GIS.* pp.710-717.

Siekierska, E.M., Pulko, S. (1986) Canada's Electronic Atlas. *Proceedings, Auto Carto London,* Vol.2. pp.407-417.

Siekierska, E.M., Taylor, D.R.F. (1991) Electronic Atlases. *CISM Journal.* Vol.45,No.1. pp.11-22.

Simons, G.L. (1985) **Expert Systems and Micros: Introduction.** Manchester: NCC Publications.

Slocum, T.A. (1988) Developing an Information System for Choropleth Maps. *Proceedings, Third International Symposium on Spatial Data Handling.* pp.293-305.

Smith, T., Peuquet, D., Menon, S., Agarwal, P. (1987) KBGIS-II. A Knowledge-based geographical information system. *International Journal of GIS.* Vol.1, No.2. pp.149-172.

Smith, T.H., Green, T.R.G. (1980) **Human Interaction with Computers.** London: Academic Press.

Smith, T.R. (1984) Artificial Intelligence and its Applicability to Geographical Problem Solving. *The Professional Geographer.* 36(2). pp.147-158.

Smith, T.R., Yiang, J.E. (1991) Knowledge-based approaches in GIS. In: Maguire, D.J., Goodchild, M.F., Rhind, D.W. (eds) **Geographic Information Systems: Principles and Applications,** Vol.1. London: Longman pp.413-425.

Stadelmann, M., Lodwick, G.D. (1988) Building a knowledge base for automated cartographic feature extraction from digital imagery. *Proceedings, Eurocarto Seven.* pp.23-31.

Steels, L., Campbell, J.A. (eds) (1985) **Progress in Artificial Intelligence.** Chichester: Ellis Horwood Limited.

Stefanovic, P., Vries-Baayens, A. (1984) Classification systems, choropleth maps and the computer. *ITC Journal.* part 1. pp.52-57.

Stoms, D. (1987) Reasoning with Uncertainty in Intelligent Geographic Information Systems. *Proceedings, GIS 87.* pp.693-700.

Su, B., Zhang, H., Li, H., Zhang, X., Zhang, Y., Zhu, X., Li, J. (1993) A knowledge base system based on GIS for thematic mapping. *Proceedings, 16th International Cartographic Conference.* pp.468-477.

Swinkels, D.A.J., Lock Lee, L., Saunders, M. (1986) Comparison of Four Expert System Development Packages. *Proceedings, 2nd International Expert Systems Conference.* pp.149-155.

Takala, T. (1987) Theoretical Framework for Computer Aided Inovative Design. In: Yoshikawa, H., Warman, E.A. (eds.) **Design Theory for CAD.** Amsterdam: Elsevier Science Publishers. pp.323-338

Ten Hagen, P.W.J., Tomiyama, T. (eds.) (1987) **Intelligent CAD Systems I.** Berlin: Springer-Verlag.

Thalman, N., Claude, Y., Laporte, G., Rousseau, J. (1982) Elect / An Interactive Graphical System for the Automated Generation of Electoral Maps. *Cartographica.* Vol.19, No.1. pp.28-40.

Tomiyama, T., Ten Hagen, P.J.W. (1987) Organization of Design Knowledge in an Intelligent CAD Environment. In: Gero, J.S. (ed.) **Expert Systems in Computer-Aided Design.** Amsterdam: Elsevier Science Publishers. pp.119-147.

Tomiyama, T., Ten Hagen, P.J.W. (1987) The concept of intelligent integrated interactive CAD systems. *Report CS-R8717.* Amsterdam: Centre for Mathematics and Computer Science.

Tomlin, D. (1980) **The Map Analysis Package.** Yale Uniersity School of Forestry and Environmental Studies.

Townsend, C. (1987) **Mastering Expert Systems with Turbo Prolog.** Indionapolis: Howard W. Sams & Co.

Trigg, A.D., Gill, G.A. (1988) Canvas: A System for Improved Colour Selection for Classified Imagery and Thematic Maps. *Report No.7*, NUTIS, Reading.

Tullis, T.S. (1981) An Evaluation of Alphanumeric, Graphic, and Color Information Displays. *Human Factors.* Vol.23(5). pp.541-550.

Ullman, J.D. (1980) **Principles of Database Systems.** London: Pitman Publishing Ltd.

Unwin, D. (1981) **Introductory Spatial Analysis.** London: University Paperbacks.

Uthe, A. (1988) User-Orientated Description and Classification of Geoscientific Data. *Geologisches Jarbruch* A 104 pp.63-74.

Van Elzakker, C. (1991) Map use research and computer-assisted statistical cartography. *ITC Journal* 1991/3 pp.122-126.

Vicars, D.W., Robinson, G.J. (1986) Generalisation from Large to Medium and Small Scale Ordnance Survey Maps Using Expert Systems Techniques. *Proceedings, Auto Carto London*, Vol.2. pp.267-275.

Wang, F., Newkirk, R. (1988) A Knowledge-Based System for Highway Network Extraction. *IEEE Transactions on Geoscience and Remote Sensing.* Vol.26, No.5. pp.525-531.

Wang, J.Y., Wu, F., Wu, Z. (1993) The research on tools of the cartographic generalization expert system. *Proceedings, 16th International Cartographic Conference.* pp.221-227.

Wang, Z. (1990) A Representation Schema for Cartographic Information. *Proceedings, Fourth International Symposium on Spatial Data Handling.* pp.782-791

Wang, Z. (1992) An Expert System for Cartographic Symbol Design. *Unpublished PhD. dissertation,* ITC, The Netherlands.

Wang, Z. (1992) An intelligent interface for application of graphic elements. *Proceedings, 5th International Symposium on Spatial Data Handling.* pp.391-400.

Wang, Z. (1993) Searching for the most suitable cartographic representation in statistical mapping. *Proceedings, 16th International Cartographic Conference.* pp.137-146.

Wang, Z., Brown, A. (1991) A knowledge-based system for selection of area colours for maps. *Proceedings, 15th ICA Conference.* pp.945-949.

Waterman, D.A. (1986) **A Guide to Expert Systems.** Reading, Mass.: Addison-Wesley Publishing Company.

Waters, N.M. (1989) Expert systems within a GIS: Knowledge acquisition for spatial decision support systems. *Proceedings, Canadian National Conference on GIS.* pp.740-759.

Waugh, T.C. (1980) A Parameter Driven Language for use in Application Systems. *Harvard papers on GIS*, Volume 7.

Waugh, T.C., Carruthers, A.W. (1988) **GIMMS Reference Manual, Release 5.** Edinburgh: GIMMS (GIS) Ltd.

Weber, H.R. (1985) Meditation on Man-Machine Interfaces or Our Personal Role in Graphics Dialogue Programming. *Computers and Graphics.* Vol.9,3. pp.237-245.

Webster, C. (1990) Rule-based spatial search. *International Journal of GIS.* Vol.4, No.3. pp.241-260.

Weibel, R. (1991) Amplified intelligence and rule-based systems. in: Buttenfield, B.P., McMaster, R.B. (eds) **Map Generalization: making rules for knowledge representation.** Harlow: Longman. pp.172-186.

Weibel, R., Buttenfield, B.P. (1992) Improvement of GIS graphics for analysis and decision-making. *International Journal of GIS.* Vol.6,No.3. pp.223-246.

Weiss, S.M., Kulikowski, C.A. (1984) **A Practical Guide to Designing Expert Systems.** London: Chapman and Hall Ltd.

White, W.B., Morse, B.W. (1987) Aspenex: an Expert System Interface to a Geographic Information System for Aspen Management. *Artificial Intelligence Applications in Natural Resource Management.* Vol. 1, No. 2. pp.49-53.

Wilkinson, G.G. (1987) The search problem in automated map design. *Cartographic Journal.* Vol.24, No.1. pp.53-55.

Wilkinson, G.G., Fisher, P.F. (1987) Recent Developments in Geo-Information Systems. *Cartographic Journal.* Vol.24, No.1. pp.64-70.

Williams, C. (1986) Expert Systems, Knowledge Engineering, and AI Tools - An Overview. *IEEE Expert.* Vol.1, No.4. pp.66-70.

Williams, L.G. (1971) Obtaining Information from Displays with Discrete Elements. In: Castner, H.W., McGrath G. (eds). *Cartographica, Monograph No.2.* pp.29-34.

Wright, G., Rabbits, L. (1987) Card Tricks for Smart Players. *Expert Systems User.* February. pp.16-19.

Wu, J-K., Chen, T., Yang, L. (1989) QPF. A versatile querry language for a knowledge-based geographical information system. *International Journal of GIS.* Vol.3, No.1. pp.51-58.

Yazdani, M. (1989) Shells Versus Toolkits. *Expert Systems User.* Vol.5, No.1. pp.16-19.

Yijiang, Z. (1991) Research and realization of cartographic expert systems. *Proceedings, 15th ICA Conference.* pp.455-459.

Yoeli, P.H. (1972) The Logic of Automated Map Lettering. *Cartographic Journal.* Vol.9,No.2. pp.99-109.

Yoshikawa, H. Warman, E.A. (eds.) (1987) **Design Theory for CAD.** Amsterdam: Elsevier Science Publishers.

Zhan, F. (1991) Structuring the knowledge of cartographic symbolization - An object oriented approach. *Proceedings, AutoCarto 10.*

Zhang, W., Brooke, G., Kubik, K. (1993) Knowledge representation approach to cartographic conceptual model formalisation. *Proceedings, 16th International Cartographic Conference.* pp.137-149.

Zhang, W., Li, H., Zhang, X. (1989) Mapgen: An expert system for automatic map generalization. *Proceedings, 13th ICA Conference,* Vol.4 pp.151-157.

Zhang, W., Su, B., Li, H., Zhang, X. (1991) A Knowledge-Based Approach to Thematic Mapping. *Unpublished paper,* Wuhan Technical University of Surveying and Mapping, China.

Zhen, T., Wang, J.Y., Lin, A.S. (1993) Knowledge representation and reasoning in nautical chart design. *Proceedings, 16th International Cartographic Conference.* pp.93-102.

Zhu, X. (1994) A Framework for Intelligent Spatial Decision Support Systems. *Proceedings, GIS Research UK.* pp.280-285.

Zoraster, S. (1986) Integer Programming Applied to the Map Label Placement Problem. *Cartographica.* Vol.23, No.3. pp.16-27.

Zoraster, S. (1987) Practical Experience with a Map Label Placement Program. *Proceedings, AutoCarto 8.* pp.701-708.

APPENDICES

APPENDIX A

Program listing

```
/*****************************************************************/
/*                                                             */
/*    GLOBDOMS.PRO                                             */
/*                                                             */
/*    Global domain declarations - include in all modules      */
/*          includes Toolbox standard domains                  */
/*          includes Database predicate definitions            */
/*                                                             */
/*    D.F.  8 march 1991                                       */
/*          20/05/93    added setup database                   */
/*          04/08/93    CHANGE TO GLOBAL DECS                  */
/*                      moved some databases to local          */
/*          31/08/93    changed several dec's to SYMBOL        */
/*****************************************************************/

/*****************************************************************/
/*                                                             */
/*    FUNCTIONS   6 AUG 1993                                   */
/*                                                             */
/*    Declares global domains and database section             */
/*    included in all project modules                          */
/*    incorporates other include files - utils, toolbox, etc   */
/*    GLOBAL PREDICATES are declarations of main modules       */
/*                                                             */
/*    GLOBAL DATABASES                                         */
/*                                                             */
/*    DATA        holds attribute & coordinate data            */
/*    FRAME       knowledge frame for current map              */
/*    KBASE       main knowledge base                          */
/*    KSYMBOLRULES rules on symbolisation for screen maps      */
/*    META        meta data for each feature class            */
/*    SETUP       user setup data (setup.kba)                  */
/*    SCREENS          for toolbox screen divers               */
/*    WORK        working database                             */
/*                                                             */

GLOBAL DOMAINS
   ROW,COL,LEN,ATTR     = INTEGER
   STRINGLIST       = STRING*
%   INTEGERLIST     = INTEGER*  % replaced by CONSTANT referral to
                                % BGI_ilist 29/8/94
   REALLIST         = REAL*
   SYMBOLLIST       = SYMBOL*
   DATED            = dated(INTEGER,INTEGER,INTEGER)
   KEY              = cr; esc; break; tab; btab; del; bdel; ctrlbdel; ins;
              end; home; fkey(INTEGER); up; down; left; right;
              ctrlleft; ctrlright; ctrlend; ctrlhome; pgup; pgdn;
              ctrlpgup; ctrlpgdn; char(CHAR); otherspec
   FIELD_NAME       = SYMBOL
   TYPE             = int(); str(); real()
   SELECTOR         = STRING                % for treemenu
   TREE             = tree(STRING,SELECTOR,TREELIST)
   TREELIST         = TREE*
   PAIR             = feat(SYMBOL,INTEGER);coord(REAL,REAL);
                    ints(INTEGER,INTEGER);symbs(SYMBOL,SYMBOL)
   PAIRLIST         = PAIR*
   FILE             = INPUT;OUTPUT;DATAFILE
/* symbolspec changed 23/11/93
   SYMBOLSPEC       =      symbolspec(
                    STRING,          % feature (class or sub-class)
                    SYMBOL,          % point,line or area
```

```
                  INTEGER,    % colour
                  INTEGER,    % form
                  INTEGER,    % dimension
                  INTEGER)    % level
*/
   SYMBOLSPEC     =     symbolspec(
                  SYMBOL,           % Type - point,line or area
                  SYMBOL,           % hue
                  SYMBOL,           % lightness
                  SYMBOL,           % saturation
                  SYMBOL,           % form code
                  SYMBOL,           % form
                  INTEGER,          % orientation
                  SYMBOL,           % dimension
                  INTEGER)          % level
   SYMBOLSPECLIST = SYMBOLSPEC*

CONSTANTS
INTEGERLIST = BGI_ILIST

/*************************************************************/
/*    Database for toolbox and utilities             */
/*************************************************************/
GLOBAL DATABASE - SCREENS

/* Database declarations used in SCRHND */
   insmode
   actfield(FIELD_NAME)
   screen(SYMBOL,SCREENS)
   value(FIELD_NAME,STRING)
   field(FIELD_NAME,TYPE,ROW,COL,LEN)
   txtfield(ROW,COL,LEN,STRING)
   windowsize(ROW,COL)
   notopline

/*************************************************************/
/* Database declarations for setup file - "setup.kba"      */
/*************************************************************/
GLOBAL DATABASE - SETUP
/*    (type of file, name) */
   flname(SYMBOL,STRING)

/* types of file are:
      kbasefile   name of knowledge base for descript,
                        locate & select
      locatefile  name of file with basic location info
                        for use in determining limits
      repfile           name of knowledge base of representations

      symbolfile  name of knowledge base for symbolisation

      plotfile    name of knowledge base for plot translations
                        from general to specific media
      metafile    name of meta data file

      datapath    path to location of data files
*/

/*************************************************************/
/* Database declarations for knowledge base                  */
/*************************************************************/
```

```
GLOBAL DATABASE - KBASE
/* GENERAL */

/* rule(name,inlist1,inlist2,answer)*/
   krule(SYMBOL,SYMBOLLIST,SYMBOLLIST,SYMBOL)

/* required(context,by,what,probability)  context gives qualification
                                   probability -5 to +5    */
   krequired(SYMBOL,STRING,STRING,INTEGER)

/* member_of(main class, sub class) relates sub classes to main items
                             to allow lists to be formed */
   kmember_of(STRING,STRING)

/* some lists */

/*base_info       base info types*/
/*theme_info      thematic info types*/
/*scales          rounded scales*/
   klist(SYMBOL,STRINGLIST)

/* map_content(map_type,basic_type,[base info scores],
            [thematic info],comment) */
   kmap_content(STRING,STRING,INTEGERLIST,STRINGLIST,STRING)

/* DESCRIPTION */

/*klevel_of_detail(user,purpose,media,value)*/
   klevel_of_detail(SYMBOL,SYMBOL,SYMBOL,INTEGER)

/* LAYOUT */

/* scale_range(lower, upper)  range of scales handled by system
                        values set in kbase for flexibility */
   determ kscale_range(REAL,REAL,SYMBOL)

/* SELECTION */

/* kbase_select(scale range,level of detail,purpose, min score)
      set of rules about minimum score for selection of base info from
      known scale, l of d and purpose.  ie returns score (kselect_index) */
/*determ kbase_select(SYMBOL,SYMBOL,SYMBOL,INTEGER)    */

/**************************************************************/
GLOBAL DATABASE - KLOCATE

/* extent(Name,Part_of,east,west,south,north)   extent of area in lat & long
*/
kextent(STRING,STRING,REAL,REAL,REAL,REAL)      % used for textual description
                                    % of area to be mapped

/***************************************************************/
GLOBAL DATABASE - KREPS

/* SYMBOLISATION */

/* subclass(main class, sub class)  relates sub classes to main classes
                              of features for assigning reps      */
   ksubclass(STRING,STRING)

/* representation_type(number,description)*/
```

```
   krepresentation_type(INTEGER,SYMBOL)

/* rep(context,by,what,probability) context gives qualification
                                    probability -5 to +5    */
   krep(SYMBOL,STRING,STRING,INTEGER)

/* conflict(type, value, value) */
   kconflict(SYMBOL,SYMBOL,SYMBOL)

/*****************************************************************/
GLOBAL DATABASE - KSYMRULE

   ksymbolspec(
            STRING,                 % feature class
            STRING,                 % feature
            SYMBOLSPEC)

   kassociate_with(     STRING,           % feature 1
                  STRING,           % feature 2
                  SYMBOL,           % property
                  INTEGER)    % probability

   kconvention(         STRING,           % feature_class
                  STRING,           % feature
                  SYMBOL,           % property
                  SYMBOL,           % value
                  INTEGER)    % probability

   ksymbolset(                % sets of defined symbols
            SYMBOL,                 % name
            SYMBOL,                 % type
            STRING,                 % description
            SYMBOLSPECLIST)         % list of values

   ksymbollist(               % list of defined symbol elements
                  % eg hues, shapes
                  SYMBOL,           % name
                  SYMBOLLIST) % list of values

   kstringlist(               % lists of names for feature, classes, etc
            SYMBOL,                 % name of list
            STRINGLIST)       % list of names

/* symbol conflicts */
   ksymbol_conflict(
            SYMBOL,           % media
            INTEGER,          % background colour
            SYMBOL,           % type of conflict - PLA
            INTEGER,          % first variable
            INTEGER)          % second variable

/* next 2 commented out 23/11/93 for symrule2
/* symbol_spec(media,info_name,relate_to,rep_type,rep_num,
            colour_lut,size_lut,form_lut) */
   ksymbol_spec(SYMBOL,STRING,STRING,STRING,STRING,SYMBOL,SYMBOL,SYMBOL)


/* lists of pre defined sets of symbols */
   ksymbolset(    SYMBOL,                 % output media
            INTEGER,          % background colour (see constants)
            SYMBOL,                 % name of set
            INTEGERLIST)            % list of codes (constants)
*/

   klevel(              % places features on levels for plotting
            SYMBOL,                 % feature
            SYMBOL,                 % rep type
            INTEGER)          % level

/****************************************************************/
/*    Database for current run                      */
/*    The basic frame of knowledge representation         */
/****************************************************************/
GLOBAL DATABASE - FRAME

/* GENERAL */

/* current_map(number) */
   determ fcurrent_map(INTEGER)
/* map_date(date)             date map designed or modified*/
   determ fmap_date(DATED)
/* map_author(name)                 who made the map*/
   determ fmap_author(STRING)
/* map_title(title)                 a simple title for identification*/
   determ fmap_title(STRING)
   determ fnotes(STRING)

/* DESCRIPTION */
/* map_class(name)                  class map type belongs to */
   determ fmap_class(STRING)
/* map_type(name)                   Type of map*/
   determ fmap_type(STRING)
/* map_purpose(purpose)             OVERVIEW, ANALYSIS*/
   determ fmap_purpose(SYMBOL)
/* map_user(SYMBOL)                 who is the intended user
                              AUTHOR, GENERAL USER, KNOWLEDGABLE*/
   determ fmap_user(SYMBOL)
/* output_media (media)             screen, paper, overhead, slide */
   determ foutput_media(SYMBOL)
/* level_of_detail                  Value between 1 (low) & 10 (high)*/
   determ flevel_of_detail(REAL)

/* LAYOUT */
/* scale(rep. fraction)             actual scale of map */
   determ fscale(REAL)
/* scale_type(code)           scalelarge
                              scalemedium
                              scalesmall
                              values set by kscalerange in kbase  */
   fscale_type(SYMBOL)
/* format (width,height) */
   determ fformat(REAL,REAL)
/* lat_long(left, right, bottom top)      location in lat long*/
   determ flat_long(REAL,REAL,REAL,REAL)
/* limits(left,bottom,right,top)    coords in data base system*/
   determ flimits(REAL,REAL,REAL,REAL)


/* SELECTION */
/* selection index(value) */
   determ fselect_index(INTEGER)
/* base_info_list([list of base info to include]) */
```

```
   determ fbase_info_list(STRINGLIST)
/* base_info(topic,score)*/
   fbase_info(STRING,INTEGER)
/* theme_info_list([list of thematic info to include]) */
   determ ftheme_info_list(STRINGLIST)

/*SYMBOLISATION */
/* representation(FEATURE,REPNUM)*/
   frepresentation(STRING,STRING)
   fcategories(   STRING,                  % feature class
            STRINGLIST)       % list of catagories (or classes)
                              % to include
   fclass_intervals(STRING,         % Feature class
            REALLIST)         % list of class breakes
                              % (number of classes + 1)
   determ fbackground_colour(INTEGER)      % colour of background
   fsymbolism(    STRING,                  % feature class
            STRING,                 % category or class
            SYMBOLSPEC)       % symbol

/*****************************************************************/
/*                                                      */
/*   META DATA                                          */
/*                                                      */
/*****************************************************************/
GLOBAL DATABASE - META

% data_description
% one entry per feature_class
   kmeta_data(          % describes the nature of the spatial data
               % and provides links to attributes & coords.

            SYMBOL,           % feature category
            DATED,            % date digitised
            STRING,           % comments on source

            SYMBOL,           % phenomena discrete - point at map scale
                              %     linear
                              %     specific_area
                              %     continuous_surface
            SYMBOL,           % spatial data    point
                              %     line
                              %     boundary (polys not explicit)
                              %     polygon
                              %     cells
            SYMBOL,           % attribute level of measurement of spat. data
                              %     identical
                              %     feature_coded
                              %     hierarchical_feature_coded
                              %     ordinal
                              %     interval
                              %     ratio_abs
                              %     ratio_den
                              %     external (depends on
                              %        data linked to)
            SYMBOL,           % nature    tangible
                              %     conceptual
            SYMBOL,           % symbolic name of coordinate file
            STRING,           % look up table of feature codes, etc.
            SYMBOL,           % symbolic name of associated datafile
            SYMBOL            % symbolic name in datafile
            )

% one entry per coordinate file
   kcoord_file(                % meta data about coordinate file
            SYMBOL,                  % symbolic name for file
            SYMBOL,                  % file type
                              %     point
                              %     line
                              %     boundary
                              %     polygon
                              %     cell
            DATED,                   % digitising date
            REALLIST,         % limits of data
            REAL,             % Digitising scale
            STRING,                  % source
            SYMBOL,                  % projection
                              %     sphere (Lat & Long)
                              %     plane
                              %     projection name
            SYMBOL,                  % units
                              %     degrees
                              %     kilometers
                              %     miles
            REAL,             % scale factor
                              % multiplier of file coords
                              %     to get units
            STRING,                  % filename
            STRING                   % comments
            )

   klook_up(                  % lookup table to relate kbase names to
                        % names (feature codes) in data file
            SYMBOL,                  % Feature class
            SYMBOL,                  % name in kbase
            SYMBOL                   % name in data file
            )

   kdata_file(                % information about data file
            SYMBOL,                  % symbolic name of file
            SYMBOL,                  % file type
                              %     data - data only
                              %     coord - mixed data & coords
            DATED,                   %
            STRING,                  % source of data
            SYMBOLLIST,       % names, in order, of variables
                              % kmeta_data gives properties
            STRING,                  % filename
            STRING)                  % comments

/*** note ***/
/*    the coord_file & data_file given in meta may be the same when
      file contains both coords & data.
      The system checks for this and only opens one file
*/

GLOBAL DATABASE - PLOTDATA    % the database of coordinates and data

   node(
            INTEGER,          % seq no.
            SYMBOL,                  % feature code
            INTEGER,          % xcoord
            INTEGER,          % ycoord
            SYMBOL,                 % name
            REAL)             % value
   chain(
            INTEGER,          % seq no
            SYMBOL,                 % feature code
            SYMBOL,                 % right area
            SYMBOL,                 % left area
            INTEGER,          % npts
            INTEGER,          % start node
            INTEGER,          % end node
            INTEGERLIST)            % coords
   polygon(
            INTEGER,          % seq no
            SYMBOL,                 % name / feature code
            INTEGER,          % npts
            REAL,             % area
            INTEGER,          % x centroid
            INTEGER,          % y centroid
            INTEGERLIST)            % coords

   data(                % currently fixed format
                        % 3 data values
            INTEGER,          % seq no
            SYMBOL,                 % name or feature code
            REAL,             % var1
            REAL,             % var2
            REAL)             % var3

/*************************************************************/
/*    Working database for current run               */
/*    stores flags, intermediate values, etc.             */
/*************************************************************/
GLOBAL DATABASE - WORK
/* knowledge base name  */
   wkbase_name(SYMBOL)
/* reselect(flag)             flag for to relselection*/
   wreselect(SYMBOL)
/* contents(list)             list of map contents */
   wcontents(STRINGLIST)
/* quit("y" or "n")                 flag for quit */
   wquit(SYMBOL)
% misc lists
   wlist(SYMBOL,STRINGLIST)
   wintegerlist(SYMBOL,INTEGERLIST)
   wreallist(SYMBOL,REALLIST)
% flag from check_symbol to show reason for error
   determ wcheck_symbol(SYMBOL)
% symbol elements for determined by rules for features
% got by get_symbols
   wsymbolset(    STRING,                 % feature_class
            SYMBOL,                 % code for list type
            INTEGERLIST)            % list of values


% number of screen pixels in window, or number of 'dots' available
% in plot window
   determ wtotpix(REAL)

   non_determ wcolour(        % used for unclassed area (polit.pro)
            SYMBOL,                 % zone
            SYMBOL)                 % class

/*****************************************************************/
/*                                                             */
/*    GLOBAL PREDICATES                                        */
/*                                                             */
/*****************************************************************/
GLOBAL PREDICATES
/*    These are called by MAIN or other modules and
      represent the major modules and sub modules
      All calls within these modules are local
      (formerly handled by include statements)
      descriptions are to be found locally      */
%   load_frame              % added 01/09/94
   description
   layout
   selection
   symbolisation
      getsymbols
      check_symbol(SYMBOLSPEC,SYMBOLSPECLIST) - (i,i)
      assign_levels
   screenplot
      sort_adjacent_classes(SYMBOL) - (i)
   modify

/**********************************/
/*    FOR GRAPHICS TOOLBOX (from gutil.pre)     */
include "tools\\graph.pre"

/*****************************************************************/
/*    UTILITIES ETC REQUIRED BY ALL MODULES                    */
/*****************************************************************/
/* Includes moved to individual modules to save code space 27/10/93
      only those specifically needed by module included
   include "tools\\tpreds.pro"
   include "tools\\status.pro"
   include "tools\\menu.pro"
   include "tools\\longmenu.pro"
   include "tools\\lineinp.pro"
   include "tools\\filename.pro"
   include "utils.pro"
   include "tools\\scrhnd.pro"
   include "tools\\tree.pro"        % now only called by descript
/************************************************************/
/* DEFAULT VALUES FOR SCRHND PREDICATES                    */
/************************************************************/
CLAUSES
   field_action(none).
   field_value(Fname,Val):- value(Fname,Val),!.
   noinput(none).
*/
```

```
/****************************************************************/
/*                                                          */
/*    MAIN PROGRAM - MAIN.PRO       12 DEC 1989             */
/*                                                          */
/****************************************************************/

/****************************************************************/
/*                                                    */
/*    FUNCTIONS   4 AUG 93                            */
/*                                                    */
/*    INTIALISE   makes welcome screen                */
/*                loads knowledge base                */
/*                loads screen driver                 */
/*    CLOSEDOWN   closes down screens                 */
/*                saves maps                          */
/*    MAIN        presents main menu                  */
/*                gets choice of activity             */
/*    new_map     clears frame & increments map number      */
/*    design_map  calls sequence of predicates to fill      */
/*                 frame                              */
/*                                                    */

/****************************************************************/
/*                                                          */
/*    MODIFICATIONS                                         */
/*                                                          */
/*       01/91    remove declarations to GDOMS              */
/*    08/04/91    change to new database predicates         */
/*    27/07/92    all files now on C:                       */
/*    20/05/93    introduced setup file for filenames etc   */
/*    04/08/93    major rewrite into modules - V. 0.3       */
/*    01/09/94    added load_frame                          */

project "mapdes"
include "globdoms.pro"
   include "tools\\tpreds.pro"
   include "tools\\status.pro"
   include "tools\\menu.pro"
%   include "tools\\longmenu.pro"
   include "tools\\lineinp.pro"
   include "tools\\filename.pro"
   include "utils.pro"
   include "tools\\scrhnd.pro"
%   include "tools\\tree.pro"             % now only called by descript

/*************************************************************/
/* DEFAULT VALUES FOR SCRHND PREDICATES                    */
/*************************************************************/
CLAUSES
   field_action(none).
   field_value(Fname,Val):- value(Fname,Val),!.
   noinput(none).

/************************************************************/
/*    Database of maps created in current run             */
/************************************************************/
DATABASE - KMAPS
/* map(number,title,date,author,
      type,purpose,user,output_media,level_of_detail,notes,
      left,bottom,right,top,width,height,scale,
      content,symbolism)      */
```

```
kmap(INTEGER,STRING,DATED,STRING,
      SYMBOL,SYMBOL,SYMBOL,SYMBOL,INTEGER,STRING,
      REAL,REAL,REAL,REAL,REAL,REAL,REAL,
      STRINGLIST,STRINGLIST)

/*************************************************************/
/*          INITIALISE                         */
/*************************************************************/
PREDICATES
   initialise                % sets up windows etc
   welcome                   % writes message on screen
   load_setup                % loads set up file
   check_kbase               % finds and test loads kbases
   check_kbasel              % working pred
           (SYMBOL)          % name of kbase
   load_screens              % loads screen driver

CLAUSES
   initialise:-
      makewindow(1,7,7,"Messages",20,0,4,80),
      makewindow(2,31,7,"MAPDESIGNER",0,0,20,80),
      write("Please wait ... "),
      welcome,
      load_setup,!,
      check_kbase,!,
      load_screens,!.

   initialise:-failed("initialise").

   welcome:-
      clearwindow,
      nl,nl,
      write ("                        Welcome to"),
      nl,nl,
      write("                         MAPDESIGNER"),
      nl,nl,
      write("                    Map Design Expert System"),
      nl,nl,
      write("                 A Computer Program to Aid Map Design."),
      nl,nl,nl,
      write("                     Version 0.4, May 1994"),
      nl,nl,nl,nl,
      write("                   Copyright:  David Forrest "),
      makestatus(112,
"                            PRESS ANY KEY TO CONTINUE" ),
      readchar (_).

   load_setup:-
      message("Loading setup information",""),
      file_exist("setup.kba"),
      message("set up  found, now loading",""),
      consult("setup.kba",setup),!,
      message("setup loaded","").
   load_setup:-
      file_exist("setup.kba"),!,
      ermessage("Error loading setup.kba",
                "Error in knowledge structure"),
      fail.
   load_setup:-
      ermessage("Error loading setup","File  setup.kba  not found"),
      fail.
```

```
check_kbase:-
   message("Checking Knowledge Bases ",""),
   check_kbase1(kbasefile),!,
   check_kbase1(locatefile),!,
   check_kbase1(repfile),!,
   check_kbase1(symbolfile),!.

check_kbase1(kbasefile):-
   flname(kbasefile,Filename),
   file_exist(Filename),
   consult(Filename,kbase),!,
   message("Knowledge loads ok",kbase),
   retractall(_,kbase).
check_kbase1(locatefile):-
   flname(locatefile,Filename),
   file_exist(Filename),
   consult(Filename,klocate),!,
   message("Knowledge loads ok",klocate),
   retractall(_,klocate).
check_kbase1(repfile):-
   flname(repfile,Filename),
   file_exist(Filename),
   consult(Filename,kreps),!,
   message("Knowledge loads ok",kreps),
   retractall(_,kreps).
check_kbase1(symbolfile):-
   flname(symbolfile,Filename),
   file_exist(Filename),
   consult(Filename,ksymrule),!,
   message("Knowledge loads ok",ksymrule),
   retractall(_,ksymrule).
check_kbase1(plotfile):-
   flname(plotfile,Filename),
   file_exist(Filename),
   consult(Filename,kplot),!,
   message("Knowledge loads ok",kplot),
   retractall(_,kplot).
check_kbase1(Kbname):-
   flname(Kbname,Filename),
   file_exist(Filename),!,
   ermessage("Error loading knowledge base",
             Kbname),
   fail.
check_kbase1(Kbname):-
   flname(Kbname,Filename),
   ermessage("Error loading knowledge base, file not found",
   Filename),
   fail.

load_screens:-
   message("Loading screen driver ",""),
   file_exist("screens.scr"),
   consult("screens.scr",screens),!,
   message("Screen driver loaded","").
load_screens:-
   ermessage("Error in screen driver","or 'screens.scr' not found"),
   fail.


/***************************************************************/
```

```
/*          CLOSEDOWN                                  */
/**************************************************************/
PREDICATES
   closedown
   removescreens              /* removes data for screen layouts */
%   quit                            /* prompt if user wants to quit */

CLAUSES
   closedown:-
      clearwindow,nl,nl,nl,nl,nl,
      write("           MAP DESIGN EXPERT SYSTEM TERMINATING"),
      nl,nl,
      write("                We hope you were successful"),
      nl,nl,
      pause,
      status(9),
      message("Performing orderly shutdown",""),
      shift_screen(nul),
      removescreens,
      retract(fcurrent_map(_)),
% close files, print diags etc.
      save("kmap.dat",kmaps),       % save updated map specs
% save details of current map to separate file if required
      message("map definitions saved in kmap.dat",""),
      save("frame.frm",frame),
      message("current frame saved in frame.frm",""),
      pause,
      clearmessage,
      makewindow(1,0,0,"",0,0,25,80).

   removescreens:-
      retract(screen(_,_)), fail.
   removescreens.

/***************************************************************/
/***************************************************************/
/*                                                           */
/*          Main Program                                     */
/*                                                           */
/***************************************************************/
/***************************************************************/
PREDICATES
   mainmenu             % main controling menu
   mainmenul(INTEGER)         % action for above
   design_map                 % calls modules to design map
   new_map              % initialises new (subsequent) map
   utilities                  % enters utitities menu
   storemap             % stores map design specs

CLAUSES
   mainmenu:-                       /* repeat menu until get */
      repeat,                       /* request to quit (choice=8) */
      shiftwindow(2),
      clearwindow,
      status(0),
      clearmessage,
      menu(3,20,7,7,[" Design a new map ",
                       " Modify or display a previous map ",
                 " Load existing map design file ",
                       " Save map design ",
                       " How Map Designer Works ",
                        " Utilities ",
                        " Exit Map Designer (QUIT) "],
           " MAIN MENU ",1,Choice),
     mainmenu1(Choice),
     Choice = 7,!.

     mainmenu1(0):-!.
     mainmenu1(1):-!,new_map.
     mainmenu1(2):-!,nl,write("facility not yet available"),pause.
     mainmenu1(3):-!,
%     load_frame.
                    nl,write("facility not yet available"),pause.
     mainmenu1(4):-!,nl,write("facility not yet available"),pause.
     mainmenu1(5):-!,nl,write("facility not yet available"),pause.
     mainmenu1(6):-!,utilities.
     mainmenu1(7):-!,nl,write(
     " Are you sure you want to leave Map Designer? (y/n) "),
        readchar(T), T='y'.

% mainmenu = 1
     new_map:-                        /* clear FRAME of old map */
        fcurrent_map(N),!,
        nl,write(N),
        retractall(_,work),
        retractall(_,frame),
        Newmap = N + 1,
        asserta(fcurrent_map(Newmap)),!,
        design_map.
     new_map:-                        /* no old map */
        retractall(_,work),!,
        retractall(_,frame),
        asserta(fcurrent_map(1)),!,
        design_map.

     design_map :-              /* main clause for designing maps */
        description,!,
        layout,!,
        selection,!,
        symbolisation,!,
           getsymbols,!,
        screenplot,!,
        modify,!,
        storemap.                     /* transfer current values to kmap */
     design_map:-
       ermessage ("closing down map designer due to failure","").

     storemap:-
        fcurrent_map(N),!,
        fmap_title(Title),!,
        fmap_date(Date),!,
        fmap_author(Author),!,
        fmap_type(Type),!,
        fmap_user(User),!,
        fmap_purpose(Purpose),!,
        foutput_media(Media),!,
        flevel_of_detail(Detail),!,
        fnotes(Notes),!,
        flat_long(West,East,South,North),!,
        fformat(W,H),!,
        fscale(S),!,
        fbase_info_list(BASE),!,
        ftheme_info_list(THEME),!,
        append(BASE,THEME,ALL),
        asserta(kmap(N,Title,Date,Author,Type,User,Purpose,Media,Detail,Notes,
                          West,East,South,North,W,H,S,ALL,[])).
     storemap:-                         /* map incomplete */
        fcurrent_map(N),!,
        assertz(kmap(N,"incomplete",dated(0,0,0),"","","","","",0,"",
                    0,0,0,0,0,0,0,[],[])).

  /*********************************************************************/
  /*********************************************************************/
  /*  GOAL SECTION                                       */
  /*********************************************************************/
  /*********************************************************************/
  goal

     initialise,!,
     mainmenu,
     closedown.

  /*********************************************************************/

  /*********************************************************************/
  /*    UTILITIES SECTION                                */
  /*********************************************************************/
  PREDICATES
  %   utilities           declared in MAIN
     utilmenu
     utilmenu1(INTEGER)

  CLAUSES
     utilities:-
        utilmenu.

     utilmenu:-                         /* repeat menu until get */
        repeat,                         /* request to quit (choice=8) */
        clearwindow,
        status(0),
        clearmessage,
        menu(3,20,7,7,[" Edit Set Up file ",
                         " Display or Modify knowledge base ",
                   " Load a different knowledge base ",
                         " Load a different database ",
                         "  ",
                         " Operating System Shell ",
                         " Exit Utilities "],
           " UTILITIES ",7,Choice),
     utilmenu1(Choice),
     Choice = 7,!.

     utilmenu1(0):-!.
     utilmenu1(1):-!,nl,write("facility not yet available"),pause.
     utilmenu1(2):-!,nl,write("facility not yet available"),pause.

     utilmenu1(3):-!,nl,write("facility not yet available"),pause.
  /*   utilmenu1(3):-
        clearwindow,!,
        % get file name from user (menu?)
        readfilename(3,10,7,7,"kba","kbase.kba", Newname, existing_file),
        retractall(_,kbase),            % clear exisiting kbase
        flname(kbasefile,Oldkbase),
        asserta(flname(kbasefile,Newname)),
        load_kbase,             % load new kbase
        retract(flname(kbasefile,Oldkbase)).      % remove old kbase name
     utilmenu1(3):-!,                 % reload default
        retractall(_,kbase),
        retractall(flname(kbasefile,_)),
        assert(flname(kbasefile,"kbase.kba")),
        load_kbase.
  */
     utilmenu1(4):-!,nl,write("facility not yet available"),pause.
     utilmenu1(5):-!,nl,write("facility not yet available"),pause.
     utilmenu1(6):-!,system("",1,_).
     utilmenu1(7):-!.

  /*** end of utilities ***/
```

```
/****************************************************************/
/*          2 DESCRIPTION                                   */
/****************************************************************/
/*                                                          */
/*    MODIFICATIONS                                         */
/*    27/1/91     major changes to reflect frame            */
/*                structure                                 */
/*    30/1/91     tree menu added for map type select.      */
/*    08/04/91    renamed database predicates used          */
/*    04/08/93    change to modular prog                    */
/*    24/05/94    moved consult kbase to description        */

project "mapdes"
include "globdoms.pro"
   include "tools\\tpreds.pro"
   include "tools\\status.pro"
   include "tools\\menu.pro"
   include "tools\\longmenu.pro"
   include "tools\\lineinp.pro"
   include "utils.pro"
   include "tools\\tree.pro"        % now only called by descript

/***  tree menu functions used by get map type  ***/
/*    see PDC Toolbox pp32-34.  */
PREDICATES
   collect_tree(STRING,INTEGER,TREE)
   nondeterm collect_tree1(STRING,INTEGER,TREE)

CLAUSES
   treeaction(_,_).           %required by treemenu
                        % predicate declaration in tree.pro

   collect_tree(Node,1,tree(Node,Node,[])).
%     !.                      % cut added 20/10/93 ???
   collect_tree(Node,N,tree(Node,Node,Tree1)):-
      findall(T,collect_tree1(Node,N,T),Tree1).

   collect_tree1(Node,N,Tree):-
      N1=N-1,
      kmap_content(Node1,Node,_,_,_),
      collect_tree(Node1,N1,Tree).

/*******************************************************************/
PREDICATES
%   description                     now in globdoms
   show_description
   get_date
   get_author
   get_maptitle
   get_outputmedia
   get_outputmedia1(INTEGER)
   get_maptype
   get_maptype1(SYMBOL)
   get_mappurpose
   get_mappurpose1(INTEGER)
   get_mapuser
   get_mapuser1(INTEGER)
   get_levelofdetail
%   get_levelofdetail1(INTEGER)
```

```
CLAUSES
   description:-              % check for quit
      wquit("y"),!,
      retractall(_,kbase),
      fail.
   description:-
      clearmessage,
      makewindow(20,30,2,"DESCRIPTION",0,0,20,80),
      retractall(_,kbase),
      flname(kbasefile,Filename),
      consult(Filename,kbase),
      !,
      get_date,
      get_author,
      get_maptitle,
      get_mapuser,
      get_outputmedia,
      get_maptype,
      get_mappurpose,
      get_mapuser,
      get_levelofdetail,!,
      show_description,!,
      retractall(_,kbase),
      clearmessage,
      removewindow.

   show_description:-!,
      clearmessage,
      fcurrent_map(N),
      fmap_date(dated(Year,Month,Day)),
      fmap_author(Author),
      fmap_title(Title),
      fmap_type(Type),
      fmap_purpose(Purpose),
      fmap_user(User),
      foutput_media(Media),
      flevel_of_detail(Level),
      clearwindow,
      write("Map Number : ",N),nl,nl,
      write("     The following description of the map has been recorded"),
      nl,nl,
      write("          The brief title of the map is       : ",Title),nl,nl,
      write("          Map Author                : ",Author),nl,
      write("          Date designed                  : ",
            Day,"-",Month,"-",Year),nl,nl,
      write("          The map type is                : ",Type),nl,
      write("          The final output will be to    : ",Media),nl,nl,
      write("          The purpose of the map is for       : ",Purpose),nl,
      write("          The intended user(s)           : ",User),nl,
      write("          The level of detail (1-10) is       : ",Level),nl,nl,
      pause.

   get_date:-                      /* Existing date */
      fmap_date(dated(_,_,_)),!.
   get_date:-!,
      date(Year,Month,Day),
      asserta(fmap_date(dated(Year,Month,Day))).


   get_author:-                    /* Existing Author */
      fmap_author(_),!.
```

```
get_author:-                     /* New Author */
   status(7),
   clearwindow,
   fcurrent_map(N),
   nl,write("  Map Number : ",N),
   nl,nl,nl,
   write("     Enter name of Author for this map"), nl,nl,
   cursor(10,12),readln(Author),!,
   asserta(fmap_author(Author)),
   clearwindow,
   cursor (1,1),
   write("Map Number : ",N),
   status(1).
get_author:-!,
   ermessage("You must enter an Author",
           "It can be initials"),!,
     get_author.

get_maptitle:-                   /* Existing title */
  fmap_title(_),!.
get_maptitle:-                   /* Enter Title */
   status(7),
   clearwindow,
   fcurrent_map(N),
   nl,write("  Map Number : ",N),
   nl,nl,nl,
   write("     Enter a short name or title for this map"), nl,nl,
   write("       This is for reference purposes only "),
   cursor(10,12),readln(Title),!,
   asserta(fmap_title(Title)),
   clearwindow,
   cursor (1,1),
   write("Map Number : ",N),
   status(1).
get_maptitle:-
   ermessage("You must enter a title",
           "It can be a single character"),!,
     get_maptitle.

get_maptype :- fmap_type(_),!.         /* type determined */
get_maptype :-             % using tree menu
   status(2),
   makewindow(21,30,2,"Select most appropriate map type",1,0,19,80),
   retractall(_,treemenu),
   collect_tree("map type",9,Tree),!,  % see above for predicate
               % tree root starts with "map type" in kmap_content
   repeat,
   treemenu(right,Tree,Choice,cursor,[]),!,
   status(1),
   get_maptypel(Choice),
   clearmessage,

get_maptypel("map type"):-
   message("you must select a map type",
               "more specific types are towards the right of the tree"),
   pause,!,
   fail.
get_maptypel(Choice):-
   asserta(fmap_type(Choice)).

get_mapuser :-
```

```
      fmap_user(_),!.

   get_mapuser :-
      status(2),
      menu (5,20,7,2,[" Map Author",
                        " General user(s) ",
                        " Specialist user(s) "],
         " Who is the intended map user ? ",1,Choice),
      status(1),
      get_mapuser1(Choice).

   get_mapuser1(0):-
      quit,
      fail.
   get_mapuser1(1):- asserta(fmap_user(author)).
   get_mapuser1(2):- asserta(fmap_user(general)).
   get_mapuser1(3):- asserta(fmap_user(specialist)).

   get_mappurpose :-
      fmap_purpose(_),!.
   get_mappurpose :-
      status(2),
      menu (5,20,7,2,[" General Overview ",
                        " Detailed Analysis "],
         " What will the map be used for ? ",1,Choice),
      status(1),
      get_mappurpose1(Choice).

   get_mappurpose1(0):-
      quit,
      fail.
   get_mappurpose1(1):- asserta(fmap_purpose(overview)).
   get_mappurpose1(2):- asserta(fmap_purpose(analysis)).

   get_outputmedia:-
      foutput_media(_),!.
   get_outputmedia:-
      status(2),
      menu (5,20,7,2,[" screen ",
                        " monochrome plot ",
                        " colour plot ",
                        " slide ",
                        " overhead transparency "],
         " What is the final output media ? ", 1, Choice),
      status(1),
      get_outputmedia1(Choice).

   get_outputmedia1(0):-!,
      quit,
      fail.
   get_outputmedia1(1) :- asserta(foutput_media(screen)),!.
   get_outputmedia1(_) :- nl,write("facility not available, default is
screen"),
      pause,
      asserta(foutput_media(screen)).

   get_levelofdetail :- flevel_of_detail(_),!.
   get_levelofdetail :-
      fmap_user(User),
      fmap_purpose(Purpose),
      foutput_media(Media),!,
      klevel_of_detail(User,Purpose,Media,Val),!,
      asserta(flevel_of_detail(Val)).


/**** END OF DESCRIPTION      ****/
```

```
/**************************************************************/
/*          3 LAYOUT                                          */
/**************************************************************/
/*                                                         */
/*    MODIFICATIONS                                        */
/*                                                         */
/*    08/04/91    CHANGE TO NEW DATABASE PREDICATES        */
/*    19/04/91    add place name search                    */
/*    04/08/93    move Klocate database from globdoms      */
/*    10.04/94    closedown section to save memory         */

project "mapdes"
include "globdoms.pro"
   include "tools\\tpreds.pro"
   include "tools\\status.pro"
   include "tools\\menu.pro"
   include "tools\\longmenu.pro"
   include "tools\\lineinp.pro"
   include "utils.pro"
   include "tools\\scrhnd.pro"

/**************************************************************/
/* DEFAULT VALUES FOR SCRHND PREDICATES                      */
/**************************************************************/
CLAUSES
   field_action(none).
   field_value(Fname,Val):- value(Fname,Val),!.
   noinput(none).

PREDICATES
%   layout                      Now declared in globdoms
   show_layout
%   change_layout
   get_location
   get_location1(INTEGER)
%   change_location
   get_place
   load_places
   get_place1(STRINGLIST)
   get_extent(INTEGER,STRING)
   get_extent1(STRINGLIST)
   get_max_extent(REAL,REAL,REAL,REAL,STRINGLIST,REAL,REAL,REAL,REAL)
   get_latlong
   get_limits
   calc_limits(SYMBOL)          % SYMBOL = PROJECTION
   get_format
   get_format1(INTEGER,REAL,REAL)
   check_format(REAL,REAL)
   check_format1(STRING)
%   change_format
   calc_format(REAL,REAL)
   get_scale
   get_scale1(INTEGER,REAL)
   calc_scale(REAL)
   check_scale(REAL,REAL)
%   change_scale
   close_layout                 % shuts down module retracts local data


CLAUSES
```

```
   layout:-                % check for quit
      wquit("y"),!,
      retractall(_,kbase),
      fail.
   layout :-
      flname(kbasefile,Filename),
      consult(Filename,kbase),
      makewindow(30,30,3,"LAYOUT",0,0,20,80),
      fcurrent_map(N),
      write("Map Number : ",N),
      get_location,
      get_format,
      get_scale,!,
      trace(on),
      show_layout,
      close_layout,
      retractall(_,kbase),
      clearmessage,
      removewindow.
   layout:-
      failed("layout"),
      retractall(_,kbase),
      removewindow,
      fail.

   show_layout:-
      clearmessage,
      flat_long(W,E,S,N),
      fformat(Width,Height),
      fscale(Scale),
      nl,nl,nl,
      write("     The following layout parameters have been set"),nl,nl,
      write("           Location : West longitude  : ", W),nl,
      write("                      East longitude  : ", E),nl,
      write("                      South latitude  : ", S),nl,
      write("                      North latidude  : ", N),nl,nl,
      write("           Format   : width : ",Width," cm,  height : ",
            Height," cm"),nl,nl,
      write("           Scale    :     1 : ",Scale),!,
/* insert clause to check if this is okay */
      pause.
   show_layout:-failed("show_layout"),fail.

   get_location :- flimits(_,_,_,_),!.
   get_location :- flat_long(_,_,_,_),!,
      calc_limits(_).
   get_location :-                   /* say how you want to specify it */
      status(2),!,
      menu(5,20,7,3,[" Name of country/region ",
                     " Specify Lat & Long ",
                     " Specify Coordinates "],
      " How is location to be defined ? ",1,Choice),
      status(1),
      get_location1(Choice).
   get_location:-quit,!,fail.
   get_location:-!,
      ermessage("You must enter values for the map location",""),
      pause,
      clearmessage,
      get_location.
   get_location1(0):-quit,!.
   get_location1(1):-get_place.
   get_location1(2):-get_latlong.
   get_location1(3):-get_limits.

/***  BY PLACE NAME - COORD FILE MUST EXIST     ***/
   get_place:-
      load_places,!,
      findall(Place,kextent(Place,"Africa",_,_,_,_),Places),!,
      status(4),
      menu_mult(5,20,7,3,Places,
      " Select place(s) to be mapped ",[1],Choices),!,
      status(1),
      list_from_index(Places,Choices,Out),
      get_place1(Out),
      retractall(_,klocate),
      calc_limits(_).

   load_places:-              % FILENAME FROM SETUP.KBA
                        % add option to give filename here
      message("Loading information from disk", ""),
      flname(locatefile,Flname),!,
      file_exist(Flname),
      consult(Flname,klocate),!,
      message("file loaded",Flname).
   load_places:-
      flname(locatefile,Flname),
      file_exist(Flname),!,
      concat("Error in loading  ",Flname, Mess),
      ermessage(Mess, "Error in knowledge structure"),
      fail.
   load_places:-
       ermessage("Error loading: File not found", ""),
       fail.

   get_place1([]):-!,quit.
   get_place1([Country|[]]):-!,                    % only one selected
      status(2),
      menu(5,20,7,3,[" Full extent "," Sub regions "],
                  "Full extent or region(s)?",1,Choice),
      status(1),
      get_extent(Choice,Country).
   get_place1(Countries):-          % several countries
      get_extent1(Countries).

   get_extent(0,_):-!,quit.
   get_extent(1,Place):-            % whole place
      kextent(Place,_,W,E,S,N),!,
      asserta(flat_long(W,E,S,N)).
   get_extent(2,Place):-            % regions
      findall(Region,kextent(Region,Place,_,_,_,_),Regions),!,
      status(4),!,
      repeat,
      menu_mult(5,20,7,3,Regions,
      " select places to be mapped ",[],Choices),
      list_from_index(Regions,Choices,Out),
      status(1),
      get_extent1(Out),!.

   get_extent1([]):-!,
      ermessage("at this stage you must select somewhere to be mapped",""),
      fail.
   get_extent1([State|[]]):-!,            % single state
      kextent(State,_,W,E,S,N),!,
      asserta(flat_long(W,E,S,N)).
   get_extent1([H|T]):-
      kextent(H,_,W,E,S,N),!,
      get_max_extent(W,E,S,N,T,MW,ME,MS,MN),
      asserta(flat_long(MW,ME,MS,MN)).

   get_max_extent(W,E,S,N,[],W,E,S,N):-!.
   get_max_extent(W,E,S,N,[H1|T],MW,ME,MS,MN):-
      kextent(H1,_,W1,E1,S1,N1),
      lesser_of(W,W1,W2),
      greater_of(E,E1,E2),
      lesser_of(S,S1,S2),
      greater_of(N,N1,N2),!,
      get_max_extent(W2,E2,S2,N2,T,MW,ME,MS,MN).

/***  BY LAT AND LONG         ***/
   get_latlong :-
      status(5),
      shift_screen(latlong),!,
      scrhnd(off,Key),
      not(Key = esc),
      value("west",West),!,
      str_real(West,W),
      value("east",East),!,
      str_real(East,E),
      value("north",North),!,
      str_real(North,N),
      value("south",South),!,
      str_real(South,S),
      clearwindow,
      asserta(flat_long(W,E,S,N)),
      calc_limits(_).

/***  BY PROJECTION COORDINATES   ***/
   get_limits :-
      status(5),
      shift_screen(limits),!,
      scrhnd(off,Key),
      not(Key = esc),
      value("left",Lefts),
      str_real(Lefts,Left),
      value("right",Rights),
      str_real(Rights,Right),
      value("top",Tops),
      str_real(Tops,Top),
      value("bottom",Bottoms),!,
      str_real(Bottoms,Bottom),
      clearwindow,
      asserta(flimits(Left,Bottom,Right,Top)).

/* CALCULATE COORDS FROM LAT & LONG */
   calc_limits("Cyl_Eq_Area"):-
      flat_long(Wlong,Elong,Slat,Nlat),!,
      rad_deg(W,Wlong),
      rad_deg(E,Elong),
      rad_deg(S,Slat),
      rad_deg(N,Nlat),
      Left = W * 6371,
      Right = E * 6371,
      Bottom = sin(S) * 6371,
      Top = sin(N) * 6371,
      asserta(flimits(Left,Bottom,Right,Top)).
   calc_limits(_):-
     failed("calc_limits"), fail.

/***  FORMAT      ***/
   get_format :-             % Format in frame & ok
      fformat(Width,Height),!,
      check_format(Width,Height).
   get_format :-             % format supplied, but too large
      fformat(_,_),!,
      retract(fformat(_,_)),!,
      fail.                  % get new format
   get_format :-!,                       /* Get format */
      status(2),
      menu(5,20,7,3,[   " Full A3        - 42 * 29 cm",
                       " A3 with border - 38 * 25 cm",
                       " Full A4        - 29 * 21 cm",
                       " A4 with border - 25 * 18 cm",
                       " Full A5        - 21 * 15 cm",
                       " A5 with border - 18 * 12 cm",
                       " Fill Screen",
                       " Specify own dimensions",
                       " Calculate from Scale and Location "],
      "Select Format - Size of Map",7,Choice),
      status(1),
      get_format1(Choice,Width,Height),
      asserta(fformat(Width,Height)).

   get_format1(0,_,_):- quit,fail.
   get_format1(1,Width,Height):-
      flimits(Left,Bottom,Right,Top),!,
      (Right-Left) >= (Top-Bottom),
      Width = 42, Height = 29.
   get_format1(1,Width,Height):- Width = 29, Height = 42.
   get_format1(2,Width,Height):-
      flimits(Left,Bottom,Right,Top),!,
      (Right-Left) >= (Top-Bottom),
      Width = 38, Height = 25.
   get_format1(2,Width,Height):- Width = 25, Height = 38.
   get_format1(3,Width,Height):-
      flimits(Left,Bottom,Right,Top),!,
      (Right-Left) >= (Top-Bottom),
      Width = 29, Height = 21.
   get_format1(3,Width,Height):- Width = 21, Height = 29.
   get_format1(4,Width,Height):-
      flimits(Left,Bottom,Right,Top),!,
      (Right-Left) >= (Top-Bottom),
      Width = 25, Height = 18.
   get_format1(4,Width,Height):- Width = 18, Height = 25.
   get_format1(5,Width,Height):-
      flimits(Left,Bottom,Right,Top),!,
      (Right-Left) >= (Top-Bottom),
      Width = 21, Height = 15.
   get_format1(5,Width,Height):- Width = 15, Height = 21.
   get_format1(6,Width,Height):-
      flimits(Left,Bottom,Right,Top),!,
      (Right-Left) >= (Top-Bottom),
      Width = 18, Height = 12.
   get_format1(6,Width,Height):- Width = 12, Height = 18.
   get_format1(7,Width,Height):- Width = 18, Height = 16.5.
   get_format1(8,Width,Height):-
/* check for changestatus  */
      lineinput_leave(10,5,35,7,7,"Enter width of map in cm  : ","",Widths),
      str_real(Widths,Width),!,
      lineinput_leave(12,5,35,7,7,"Enter height of map in cm : ","",Heights),
      str_real(Heights,Height),
      removewindow,removewindow .
   get_format1(9,Width,Height):-          /*already got scale*/
      calc_format(Width,Height),!.
   get_format1(9,Width,Height):-!,        /*get scale first*/
      get_scale,
      calc_format(Width,Height).

   check_format(_,Hin):-
      flimits(_,Bottom,_,Top),
      fscale(Scale), !,
      Minheight = (Top - Bottom) / (Scale / 100000),
      Minheight > Hin,
      lineinput(10,5,60,7,7,
        " Location too high for format. Adjust Scale? (Y/N) ","Y",Ans),
      check_format1(Ans),!,
      fail.
   check_format(Win,_):-
      flimits(Left,_,Right,_),
      fscale(Scale), !,
      Minwidth = (Right - Left) / (Scale / 10000),
      Minwidth > Win,
      lineinput(10,5,60,7,7,
        " Location too wide for format. Adjust Scale? (Y/N) ","Y",Ans),
      check_format1(Ans).

   check_format1("Y"):-!,
      calc_scale(Scale),
      retract(fscale(_)),
      asserta(fscale(Scale)).
   check_format1(_):-
      message("select new format","").

   calc_format(Width,Height):-!,
      flimits(Left,Bottom,Right,Top),
      fscale(Scale),
      Width = (Right - Left) / (Scale / 100000),
      Height = (Top - Bottom) / (Scale / 100000).

/***  SCALE ***/
   get_scale :-
      fscale(Scale),
      flimits(_,Bottom,_,Top),
      fformat(_,Height),!,
      Minheight = (Top - Bottom) / (Scale / 100000),
      Minheight > Height,
      message("location too large for chosen format and scale,",
                  "scale being adjusted"),
      calc_scale(Scale1),
      retract(fscale(_)),
      asserta(fscale(Scale1)).

   get_scale :-
      fscale(Scale),
   flimits(Left,_,Right,_),
   fformat(Width,_),!,
   Minwidth = (Right - Left) / (Scale / 100000),
   Minwidth > Width,
   message("location too large for chosen format and scale,",
               "scale being adjusted"),
   calc_scale(Scale1),
   retract(fscale(_)),
   assertz(fscale(Scale1)).

get_scale :-
   flimits(_,_,_,_),
   fformat(_,_),!,
   calc_scale(Scale),
   asserta(fscale(Scale)).

get_scale :-
   status(2),
   menu(5,20,7,3,[" 1:  2 000 000 ",
                   " 1:  3 000 000 ",
                   " 1:  5 000 000 ",
                   " 1:  7 500 000 ",
                   " 1: 10 000 000 ",
                   " 1: 15 000 000 ",
                   "Calculate from location and format",
                   "Specify own scale "],
    "SCALE",7,Choice),
   status(1),
   get_scale1(Choice,Scale),
   asserta(fscale(Scale)).

get_scale1(0,_):-quit,fail.
get_scale1(1,Scale):- Scale = 2000000 .
get_scale1(2,Scale):- Scale = 3000000 .
get_scale1(3,Scale):- Scale = 5000000 .
get_scale1(4,Scale):- Scale = 7500000 .
get_scale1(5,Scale):- Scale = 10000000 .
get_scale1(6,Scale):- Scale = 15000000 .
get_scale1(7,Scale):- calc_scale(Scale).
get_scale1(8,Scale):-
   lineinput(14,5,40,7,7," Enter fractional component of scale - 1: ",
   "",Scales),
   str_real(Scales,Scale).

calc_scale(Scale):-
   flimits(Left, Bottom, Right, Top),
   fformat(Width,Height),!,
   S1 = (Right - Left) * 100000 / Width,
   S2 = (Top - Bottom) * 100000 / Height,
   lesser_of(S1, S2, S3),
   check_scale(S3,S4),
   nearest_larger(S4,[2000000,3000000,5000000,7500000,10000000,
               15000000],Scale).

calc_scale(0):-!,
   failed ("calc_scale").

check_scale(Scalein, 2000000):-  /* Scale too large */
   Scalein < 2000000,!,
   message(
    " Scale larger than maximun allowed, being adjusted to 1 : 2 000 000 ",
         "").
   check_scale(Scalein, 15000000):- /* Scale too small */
      Scalein > 15000000,!,
      message(
      " Scale smaller than minimum allowed, being adjusted to 1 : 15 000 000",
      " Map may be reduced in size").
   check_scale(Scalein,Scalein):-!.      /* Scale okay */

/*** MODULE CLOSEDOWN ***/
CLAUSES
   close_layout:-
      retractall(_,klocate).

/*** END OF LAY2.PRO ***/
```

```
/**************************************************************/
/*          4 SELECTION                                      */
/**************************************************************/

/****************************************************************/
/*                                                           */
/*    PROCESS                 24 AUG 93                      */
/*    1 - calculate selection index                          */
/*    2 - get base information, either from user or kbase    */
/*    3 - get theme information,   "                         */
/*    4 - check for other required info - if user selected  */
/*          question user about inclusion                    */
/*    5 - show list of information to be included and confirm  */
/****************************************************************/
/*                                                           */
/*    06 Sept 93  added ordering to theme info               */

project "mapdes"
include "globdoms.pro"
   include "tools\\tpreds.pro"
   include "tools\\status.pro"
   include "tools\\menu.pro"
   include "tools\\longmenu.pro"
   include "tools\\lineinp.pro"
%   include "tools\\filename.pro"
   include "utils.pro"
%   include "tools\\scrhnd.pro"
%   include "tools\\tree.pro"             % now only called by descript

PREDICATES
%   selection                        %now in globdoms
   calc_select_index(SYMBOL)         /* calculation of selection index */
   check_index(INTEGER,INTEGER)            /* check index in range 1-10 */
   show_selection
   check_selection(SYMBOL)
   change_selection
   change_selection1(INTEGER)
   get_base                    /* selection of base info */
   get_base1(INTEGER)
/* get_base_types(inputlist, workinglist,outputlist)
      for gettingdetailed contents when user selecting   */
%   get_base_types(STRINGLIST,STRINGLIST,STRINGLIST)
%   change_base
   change_base1(INTEGER)
/* selectlist(input list of scores, min score, input names, output list)
   takes list of base info scores for map type and makes list
   of those making min score    See TP User's Guide p.195
   also asserts name and score to frame */
   selectlist(INTEGERLIST,INTEGER,STRINGLIST,STRINGLIST)
   assert_base_info(STRINGLIST)
   remove_base_info(STRINGLIST,STRINGLIST)
%           flag    inlist     working    outlist
   check_requd(SYMBOL,STRINGLIST,STRINGLIST,STRINGLIST)
                              % works through inlist calling
                              % krequired to see if other things
                              % are required
                              % extend to take in probabilities
   get_theme                        /* selection of theme info */
   get_theme1(INTEGERLIST)
   order_theme(STRINGLIST,STRINGLIST)     % gets user to order theme info
   order_theme1(STRINGLIST,STRINGLIST,STRINGLIST)
```

```
%   change_theme
   change_theme1(INTEGER)
%   check_theme(STRINGLIST,STRINGLIST)    /* check for user info */

CLAUSES
   selection:-                      /* check for quit */
      wquit("y"),!,
      retractall(_,kbase).
   selection:-  /* map_type already known */
      flname(kbasefile,Filename),
      consult(Filename,kbase),
      status(1),
      clearmessage,
      makewindow(40,30,4," DATA SELECTION ",0,0,20,80),
      fcurrent_map(N),
      write("Map Number : ",N),
      get_base,
      get_theme,!,
      show_selection,!,
      retractall(_,kbase),
      removewindow.
   selection:-!,
      failed("selection"),
      retractall(_,kbase),
      removewindow,
      fail.

   calc_select_index(default):-!,
      fscale(S),
      flevel_of_detail(D),
      kscale_range(Max,_,_),
      A = ((Max / S) * 10) + ((D - 5) / 2),
      B = 11 - trunc(A),
      check_index(B,Index),
      asserta(fselect_index(Index)).

   check_index(In,Out):-
      In > 10,!,
      Out = 10.
   check_index(In,Out):-
      In < 1,!,
      Out = 1.
   check_index(In,In):-!.

   show_selection:-
      clearwindow,
      clearmessage,
      status(1),
      fcurrent_map(N),!,
      fbase_info_list(Base),!,
      ftheme_info_list(Theme),!,
      write
      (" Information to be included in map ",N),
      length_of(Base,Len),
      Ht = Len + 1,
      makewindow(41,31,20," Base Information ",2,40,Ht,30),
      write_a_list(Base),
      length_of(Theme,Len2),
      Ht2 = Len2 + 3,
      makewindow(42,31,20," Theme Information ",2,10,Ht2,30),
      nl,write_a_list(Theme),
      status(1),
      lineinput(16,5,28,65,31," Is this okay (y/n) ?  ","y", Txt),
      removewindow(43,1),
      removewindow(41,1),
      removewindow(42,1),
      shiftwindow(40),
      check_selection(Txt).
   show_selection:-!,
      failed("show selection"),fail.

   check_selection("y"):-!.
   check_selection("Y"):-!.
   check_selection(Txt):-
      frontchar(Txt,' ',Rest),!,
      check_selection(Rest).
   check_selection(_):-
%       change_selection.
      message("currently cannot be modified",""),
      pause.

   change_selection:-
      clearwindow,
      fcurrent_map(Num),
      write("Map Number ",Num),
      status(2),
      menu(5,20,7,4,[" No (more) changes",
                     " Add some base information",
                     " Delete some base information ",
                     " Add theme information",
                     " Delete theme information"],
            "Would you like to",5,Choice),
      change_selection1(Choice),!,
      show_selection.

   change_selection1(0):-!,
      quit,
      fail.
   change_selection1(1):-!.
   change_selection1(2):-!,
      change_base1(1).
   change_selection1(3):-!,
      change_base1(2).
   change_selection1(4):-!,
      change_theme1(1).
   change_selection1(5):-!,
      change_theme1(2).

/***   BASE INFORMATION   ***/
   get_base:-!,
      status(2),
      menu(5,20,7,4,[" User select ", " Default system selection"],
            " How is Base Information to be selected? ",2,Choice),
      status(1),
      get_base1(Choice).

   get_base1(0):-!,
      quit, fail.
   get_base1(1):-!,                  % user selection
      message("currently only auto select available",""),
      pause,
      get_base1(2).
   get_basel(2):-!,                     % auto selection
      calc_select_index(default),!,
      fmap_type(T),
      fselect_index(Score),
      kmap_content(T,_,Scores,_,_),
      klist("base_info_types",Names),!,
      selectlist(Scores,Score,Names,Out),
      check_requd(baseinfo,Out,[],Newbaselist),
      asserta(fbase_info_list(Newbaselist)).

/*  currently only auto select
   get_base_types([],Working,Working).
   get_base_types([Feat|T],Workin,Out):-
      klist(Feat,Type_list),
      status(4),
      menu_mult(2,20,7,4,Type_list,"Select sub classes",[],Choices),!,
      list_from_index(Type_list,Choices,Out_list),
      append(Workin,Out_list,Workout),!,
      get_base_types(T,Workout,Out).
   get_base_types([Feat|Tail],Workin,Out):-
      append(Workin,[Feat],Workout), !,
      get_base_types(Tail,Workout,Out).
  */

   change_basel(1):-                    % additional info
      klist("base_info_types",Base),
      status(4),
      menu_mult(5,20,7,4,Base,"Select info to add",_,Choices),!,
      list_from_index(Base,Choices,New),
      fbase_info_list(L1),
      append(L1,New,L2),
      retract(fbase_info_list(_)),
      asserta(fbase_info_list(L2)),
      assert_base_info(New).
   change_basel(2):-                    % info deleted
      fbase_info_list(L1),
      status(4),
      menu_mult(5,20,7,4,L1,"Select info to delete",_,Choices),!,
      list_from_index(L1,Choices,Out),
      retract(fbase_info_list(_)),
      asserta(fbase_info_list(Out)),
      remove_base_info(L1,Out).

   selectlist([],_,_,[]).                          % empty list
%nl,write("entered first sl clause").
   selectlist([H|T],Score,[_|T1],T2):-             % discard head
/* if H is < Score then skip over but cut head from list 1*/
      H < Score,!,
      selectlist(T,Score,T1,T2).
   selectlist([H|T],Score,[H1|T1],[H1|T2]):-       % keep head
/* else make head of list 2 same as list 1 and assert values */
      selectlist(T,Score,T1,T2),
      assertz(fbase_info(H1,H)).

   assert_base_info([]):-!.
   assert_base_info([Name|Tail]):-
      assertz(fbase_info(Name,10)),
      assert_base_info(Tail).

   remove_base_info([],_):-!.
   remove_base_info([H|T],New):-
```

```
      member(H,New),!,
      remove_base_info(T,New).
   remove_base_info([H|T],New):-
      retract(fbase_info(H,_)),!,
      remove_base_info(T,New).

   check_requd(_,[],Inlist,Outlist):-!,   % end of list
      uniquelist(Inlist,Outlist).   % check only one instance of each
   check_requd(Flag,[In|Inlist],Oldlist,Outlist):-
      findall(Reqd,krequired(Flag,In,Reqd,_),Reqdlist),!,
      append(Reqdlist,Inlist,Working),    % add anything found to list
                                    % working on to see if it needs
                                    % anything
      uniquelist(Working,Newwork),  % remove duplicates
      append(Oldlist,[In],Newlist), % add item done to output
      check_requd(Flag,Newwork,Newlist,Outlist).

/***   THEME INFORMATION   ***/

   get_theme:-                      /* no theme info */
      fmap_type(Type),
      kmap_content(Type,_,_,[],_),!,
      asserta(ftheme_info_list([])).
   get_theme:-                      /* well defined map type */
      fmap_type(Type),              /* - only one set of theme info */
      kmap_content(Type,_,_,[Theme|[]],_),!,
      asserta(ftheme_info_list([Theme])).
   get_theme:-                      /* more general definition of map */
      fmap_type(Type),              /* type - several possible themes */
      kmap_content(Type,_,_,Themes,_),!,
      status(4),
/*      append(Themes,["Author Supplied Information"],Showlist),  */
      Showlist = Themes,
      repeat,                       % repeat menu if fail
                                    % forces selection or quit
      menu_mult(2,20,7,4,Showlist,"Select Thematic Information",[],Choices),!,
/* must check later for new info to be supplied by user. */
      get_theme1(Choices),
      list_from_index(Themes,Choices,Out),
      order_theme(Out,Order),
      asserta(ftheme_info_list(Order)).

   get_theme1([0]):-!,
      quit.
   get_theme1([]):-!,
      message("No theme information selected","continuing with base map"),
      pause.                  % backcktrack here and force selection ?
   get_theme1(_):-!.

   order_theme([],[]):-!.
   order_theme([Only_one|[]],[Only_one]):-!.
   order_theme(In,Ordered):-
      menu(5,20,7,4,In,"Which information is most important?",1,Choice),
      member_from_index(In,Choice,Member),
      remove(Member,In,Rest),
      makewindow(44,31,4,"Selected so far",5,5,10,30),
      order_theme1([Member],Rest,Ordered),
      removewindow(44,1).

   order_theme1(Inlist,[],Inlist):-!.     % no more to do
%   order_theme1(Inlist,[Last|[]],Outlist):-!,  % last on list - don't ask
%      append(Inlist,[Last],Outlist).
   order_theme1(Inlist,Rest,Outlist):-
      shiftwindow(Old),
      shiftwindow(44),
      clearwindow,
      write_a_list(Inlist),
      shiftwindow(Old),
      menu(5,40,7,4,Rest,"Which is next most important?",1,Choice),
      member_from_index(Rest,Choice,Member),
      remove(Member,Rest,Working),
      append(Inlist,[Member],Newin),
      order_theme1(Newin,Working,Outlist).

   change_theme1(_).

/***  END OF SELECTION  ***/
```

```
/**************************************************************/
/*                                                            */
/*    5 SYMBOLISATION - SYMBOL.PRO                            */
/*                                                            */
/*    see also symrul.pro + levels.pro                        */
/**************************************************************/
/*    PROCESS
      1 - get list of selected features
      2 - order this list in terms of priority for showing feature
      3 - work down list assigning representation types
            initially assign prefered rep. type
            check for clash with reps previously assigned
            if clash occurs look for alternative for current
            if none backtrack
      4 - Assign specific symbols to representations
            initially assign preferred symbols/colours
            check for clash with previously assigned symbols
            if clash occurs look for alternative for current
            in none backtrack
      5 - Assign symbols to levels

*/
/*****************************************************************/
/*                                                            */
/*    27/2/91     Intial program                              */
/*    29/7/92     serious upgrade                             */
/*    28/10/92    some reorganisation                         */
/*    04/08/93    GO MODULAR                                  */
/*    24/08/93    some restructuring                          */
/*      /05/94    add LEVELS to symbolisation                 */

project "mapdes"
include "globdoms.pro"

   include "tools\\tpreds.pro"
   include "tools\\status.pro"
   include "tools\\menu.pro"
   include "tools\\lineinp.pro"
   include "utils.pro"

/***********************************************************/
/***  PROCESS 1 & 2 - ASSEMBLE & ORDER FEATURES        ***/
/***********************************************************/
PREDICATES
   assemble            % assembles ordered list of features
                       % from lists asserted in kbase
   assemble1(                % reduced detailed base info list
                       % to main classes
            STRINGLIST,      % input list
            STRINGLIST,      % working list
            STRINGLIST)      % output list

CLAUSES
   assemble:-
      fbase_info_list(In),
      assemble1(In,[],Out),
      ftheme_info_list(Theme),
      message("got base & theme info",""),
      append (Theme,Out,Contents),
      asserta(wcontents(Contents)),
      clearwindow.
   assemble1([],Result,Result):-
      !.    % end of list
   assemble1([Hin|Tin],Working,Out):-     % sub class
      ksubclass(Main,Hin),
      member(Main,Working),!,       % already in list
      assemble1(Tin,Working,Out).   % no change to working
   assemble1([Hin|Tin],Working,Out):-     % sub class, but not on list
      ksubclass(Main,Hin),!,
      append(Working,[Main],New),   % add main to list
      assemble1(Tin,New,Out).
   assemble1([Hin|Tin],Working,Out):-     % not sub class
      append(Working,[Hin],New),!,  % add to list
      assemble1(Tin,New,Out).

/**************************************************/
/***  PROCESS 3 - assign representations  ***/
/**************************************************/
PREDICATES
   assign_reps(               % recursive procedure - no tail elimination
         STRINGLIST,          % list or tail to be processed
         STRINGLIST,          % list of reps assigned so far
         STRINGLIST)          % output list of reps
   check_rep(                 % checks for clash with those aready on list
                           % if fails causes get_reps to backtrack
         STRING,              % representation
         STRINGLIST)          % REPS already assigned
   show_reps(                 % shows selected reps on screen
         STRINGLIST)          % list of representation types
   save_reps(
         STRINGLIST,          % list of features
         STRINGLIST)          % list of assigned representations

CLAUSES
   assign_reps([],Reps,Reps):-!.    % end of list - always succeeds
   assign_reps([H|T],Reps_so_far,Result):-
%       foutput_media(Media),!,     % may incorporate different knowledge
                                    %  bases for different media
      krep("rep_type",H,Rep,_),     % currently finds first - assumes
                                    % ordered list.
                                    % better to use findall and select best
                                    % or offer to user. On backtrack would
                                    % to select next onlist
      check_rep(Rep,Reps_so_far),   % check to see if clash with
                              %    existing reps
                              % if clash go back and look for
                              % alternative rep for current item,
                              % then for previous items
                              % if that fails return to selection
      append(Reps_so_far,[Rep],Update),   % if okay then append to list
      assign_reps(T,Update,Result). % process other contents

   check_rep(_,[]):-
   !.
   check_rep(Rep,[H|T]):-
      not(kconflict("rep_type",Rep,H)),
      !,
      not(kconflict("rep_type",H,Rep)),
      !,
      check_rep(Rep,T).
                  /* note that later it may be desirable to add
                  a mechanism to allow exceptions to general rules
                  for specific combinations of features or by
                  predetermining how the combination should be
                  symbolised */

/* show results */
   show_reps(Replist):-
      clearwindow,
      clearmessage,
      status(1),
      fcurrent_map(N),!,
      wcontents(Contents),!,
      write(" Representations for map ",N),
      makewindow(51,31,5," Information ",3,10,15,30),
      write_a_list(Contents),
      makewindow(52,31,5," Representation ",3,40,15,30),
      write_a_list(Replist),
      status(8),pause,
%      status(7),
%      lineinput(16,5,28,65,31," Is this okay (y/n) ?  ","y", Txt),
      removewindow,
      removewindow,
      clearwindow.
   show_reps(_):-!,
      ermessage("fail in show reps",""),
      failed("show_reps"),fail.

/* save results */
   save_reps([],_):-!.              % end of lists - always succeeds
   save_reps([H1|Contents],[H2|Reps]):-
      assertz(frepresentation(H1,H2)),
      save_reps(Contents,Reps).

/*************************************************/
/***  PROCESS FOUR - ASSIGN SYMBOLS TO CONTENTS  ***/
/*************************************************/
% all moved to symrule.pro 27/10/93

/*************************************************/
/***  PROCESS FIVE - ASSIGN LEVELS TO SYMBOLS    ***/
/*************************************************/
% all moved to LEVELS.pro 25/05/94

/***   SHOW SYMBOLS     ***/
PREDICATES
   show_symbols                 % shows symbols on screen
   show_symbols1(
            STRINGLIST) % list of map contents

CLAUSES
%   show_symbols.
% should show symbols graphically on the screen !
/*    work through list of contents
      for each feature class see if symbol for that class
      else see if sub classes & get these symbols
      need to check if point line or area
*/

   show_symbols:-
      wcontents(Features),
      !,
      show_symbols1(Features).

   show_symbols1([]):-!.
   show_symbols1([Feature_class|Features]):-
                              % single feature in class
      fsymbolism(Feature_class,Feature_class,
                  symbolspec(Type,Hue,Light,Sat,F_c,Form,Orient,Dim,_)),
      !,
      nl,write(Feature_class,";",Feature_class,";",Type,";",Hue,";",Light,";",
                  Sat,";",F_c,";",Form,";",Orient,";",Dim),
      pause,
      nl,!,
      show_symbols1(Features).
   show_symbols1([Feature_class|_]):-
                              % sub classes
      fsymbolism(Feature_class,Feature,
                  symbolspec(Type,Hue,Light,Sat,F_c,Form,Orient,Dim,_)),
      nl,write(Feature_class,";",Feature,";",Type,";",Hue,";",Light,";",
                  Sat,";",F_c,";",Form,";",Orient,";",Dim),
      fail.                   % get next subclass
   show_symbols1([_|Features]):-
      pause,
      nl,!,
      show_symbols1(Features).

/****************************************************************/
/*                                                          */
/***  MAIN MODULE FROM HERE                                 */
/*                                                          */
/****************************************************************/
PREDICATES
%   symbolisation now in globdoms
%   get_symbols   now in globdoms

CLAUSES
   symbolisation:-
      wquit("y"),!,
      retractall(_,kbase),
      retractall(_,kreps).
   symbolisation:-
      status(1),
      clearmessage,
      makewindow(50,30,5, " SYMBOLISATION ",0,0,20,80),
      flname(kbasefile,Filename),
      consult(Filename,kbase),
      flname(repfile,Fname),
      consult(Fname,kreps),
      assemble,                     % PROCESS 1 & 2
      wcontents([First|Contents]),  % start PROCESS 3
%      foutput_media(Media),!,
      message("finished assemble, assigning reps",""),
      krep("rep_type",First,Rep,_), % get first rep
      assign_reps(Contents,[Rep],Final_reps),   % assign representations
                              % output at this stage is list
                              % of reps in same order as features
      !,                      % once reps set don't backtrack
      message("reps assigned",""),
      show_reps (Final_reps),!,
      save_reps([First|Contents],Final_reps),!,
      message("reps saved",""),
      retractall(_,kreps),          % having got reps remove rules
      retractall(_,kbase),
%      retractall(wcontents(_)),
      message("end of assigning reps",""),
      removewindow.
   symbolisation:-
      nl,write("failed symbolisation - assigning reps - symbol.pro-main"),
      pause,
      retractall(_,kreps),
      retractall(_,kbase),
      removewindow(50,1).

/***  END OF SYMBOLISATION    ***/
```

```
/***************************************************************/
/*                                                          */
/*    5A SYMBOLISATION - ASSIGN SYMBOLS TO REPS             */
/*          SYMRUL.PRO                                      */
/*                                                          */
/***************************************************************/
/*    PROCESS
      1 -   check to see if any symbols already specified
      for those remaining:-
      2 -   use special rules for features if present
      3 -   use rules for rep types
      4 -   call check symbol for clashes
      5 -   show symbols - currently only textual list of specs
*/
/***************************************************************/
/*                                                    */
/*    --/08/93    initial attempt - failed            */
/*    --/10/93    second version                      */
/*                                                    */
project "mapdes"
include "globdoms.pro"
   include "tools\\tpreds.pro"
   include "tools\\status.pro"
   include "tools\\menu.pro"
   include "tools\\longmenu.pro"
   include "tools\\lineinp.pro"
   include "utils.pro"
   include "symchk2.pro"

/* dummy symbol checks for testing
PREDICATES
   check_colour(  SYMBOL,          % type p,l,a
                  SYMBOL,          % hue
                  SYMBOL,          % lightness
                  SYMBOL)          % saturation
CLAUSES
   check_colour(point,_,_,_).
   check_colour(line,_,_,_).
   check_colour(area,_,_,_).
*/

PREDICATES
%   get_symbols         % main calling routine - declared in globdoms.pro
   get_symbols1(STRINGLIST)
   get_known(                  % find symbols already specified
                               % either in frame or kbase
            STRINGLIST,             % Feature classes - contents
            STRINGLIST,             % working list
            STRINGLIST)             % list of F_Cs not known
   show_symbols(STRINGLIST)
   update_symbols(             % asserts or retract symbols
            STRING,                 % i  feature_class
            STRINGLIST,             % i  features
            SYMBOLSPECLIST)         % i  symbol specs
   symbol_known(               % searches for known symbols
            INTEGER,                % rule number
            STRING,                 % Feature class
            STRINGLIST,             % working list of features
            STRINGLIST)             % output list of features
   symbol_known1(       %
            STRING,                 % Feature_class
            STRINGLIST,             % Features
            STRINGLIST,             % working
            STRINGLIST)             % output list of unsymbolised features
   symbol_rule(               % main set of rules for assigning symbols
            INTEGER,                % rule number
            STRING,                 % representation type
            STRING,                 % current feature class
            STRINGLIST,             % list of subclasses symbolised
            SYMBOLSPECLIST)         % list of symbol specifications
                                    %  feature,type,colour,form,size,level
%           - (i,i,i,o,o)
   assign_symbols(            % assigns fully specified symbols to classes
            STRINGLIST,             % list of classes to be done
                                    %     - stops when list empty

            SYMBOLSPECLIST,         % symbols
            SYMBOLSPECLIST,         % working symbols list
            SYMBOLSPECLIST)         % output symbols
   assign_point_colour(       % assigns & checks a single colour
            SYMBOLLIST,             % hues
            SYMBOLLIST,             % lightnesses
            SYMBOLLIST,             % saturations
            SYMBOL,                 % hue
            SYMBOL,                 % lightness
            SYMBOL)                 % saturation
   assign_point_forms(        % assigns a series of forms from a list
            STRINGLIST,       % features
            SYMBOL,                 % hue
            SYMBOL,                 % lightness
            SYMBOL,                 % saturation
            SYMBOL,                 % form code
            SYMBOLLIST,             % list of shapes
            INTEGER,                % orientation
            SYMBOL,                 % size
            INTEGER,                % layer
            SYMBOLSPECLIST,         % input - start as []
            SYMBOLSPECLIST)         % output list of symbols
   assign_ranked_sizes(
            STRINGLIST,             % features
            SYMBOLLIST,             % list of sizes (often only 1)
            SYMBOLSPECLIST,         % working - start as all specified
            SYMBOLSPECLIST)         % final list of symbols
   assign_grad_sizes(         % assigns sizes to symbol classes
                              % - colour & form known

            STRINGLIST,             % classes
            SYMBOL,                 % hue
            SYMBOL,                 % lightness
            SYMBOL,                 % saturation
            SYMBOL,                 % form code
            SYMBOL,                 % form
            INTEGER,                % orientation
            SYMBOL,                 % size - indicates relative size
                                    % required, not absolute size

            INTEGER,                % level
            SYMBOLSPECLIST,         % workling list
            SYMBOLSPECLIST)         % output symbols
   assign_grad_sizes1(        % working clause
            STRINGLIST,             % classes
            SYMBOL,                 % hue
            SYMBOL,                 % lightness
            SYMBOL,                 % saturation
            SYMBOL,                 % form code
```

```
        SYMBOL,                 % form
        INTEGER,                % orientation
        SYMBOLLIST,             % sizes - indicates relative sizes
                                % required, not absolute size
        INTEGER,                % level
        SYMBOLSPECLIST,         % workling list
        SYMBOLSPECLIST)         % output symbols
assign_line_colour(        % assigns & checks a single colour
        SYMBOLLIST,             % hues
        SYMBOLLIST,             % lightnesses
        SYMBOLLIST,             % saturations
        SYMBOL,                 % hue
        SYMBOL,                 % lightness
        SYMBOL)                 % saturation
assign_equiv_colours(      % returns equally appearing symbols
        SYMBOL,                 % symbol type - p,l,a
        STRINGLIST,        % Groups,
        SYMBOLSPECLIST)         % Groupsymbols
assign_tints(              % assigns a set of tints to classes
        STRINGLIST,             % list of classes
        SYMBOL,                 % hue
        SYMBOL,                 % lightness
        SYMBOL,                 % saturation
        SYMBOLSPECLIST)         % list of symbols specified
assign_tints1(             % recursive assignment of tints
        STRINGLIST,
        SYMBOL,
        SYMBOL,
        SYMBOL,
        SYMBOLLIST,
        SYMBOLSPECLIST,         % working list
        SYMBOLSPECLIST).        % result
get_included(       % get base items for Feature_class from frame
        STRING,                 % feature_class
        STRINGLIST)             % features in order of hierarchy
get_included1(
        STRINGLIST,             % possible features
        STRINGLIST,             % working list
        STRINGLIST)             % output list
ask_symbolset(
        SYMBOL,                 % Feature_class
        SYMBOL,                 % type of symbol set
        SYMBOL)                 % selected set
ask_categories(            % ask user what categories required
        STRING,                 % feature class
        STRINGLIST,             % list of catagories
        STRINGLIST)             % categories required
ask_categories1(
        STRINGLIST,             % original categories
        STRINGLIST,             % alternative
        STRINGLIST)             % result
ask_classes(               % ask user about classes and intervals
        STRING,                 % Feature_class
        STRINGLIST)             % list of classes
askclasses1(
        STRING,                 % Feature_class
        INTEGER,                % Choice class type
        STRINGLIST,             % list of class names
        REALLIST)               % list of class breaks
ask_bipolar(
        STRING,                 % Feature_class
```

```
            STRINGLIST)             % list of class breaks
   askbipolar1(
            STRING,                 % Feature_class
            INTEGER,                % menu choice
            STRINGLIST,             % list of class names
            REALLIST)               % list of class breaks
   setclasses(
            SYMBOL,                 % code for type
            INTEGER,                % number of classes
            STRING,                 % Feature_class,
            REALLIST)               % list of class breaks
   setclasses1(               % set breaks of equal intervals
            REAL,                   % Min
            REAL,                   % Interval
            INTEGER,                % Number of classes
            REALLIST,               % input list of classes
            REALLIST)               % output list of classes
   setclasses2(               % set breaks for quantile classes
            INTEGER,                % number of classes
            REAL,                   % number in each class
            REALLIST,               % ordered list of values
            REALLIST,               % working list
            REALLIST)               % returned list of breaks
   setclasses3(               % set breaks for exploratory classes
            REALLIST,               % first set of intervals
            REALLIST,               % second set of intervals
            REALLIST,               % working list
            REALLIST)               % output list of breaks
   load_data(                 % reads in data values from file
            SYMBOL,                 % filename
            INTEGER,          % position of variable in term
            REALLIST)         % list of values
   load_data1(   REAL)
   readval( INTEGER,REAL)
   nondeterm  repeatread(     FILE)

% check symol clauses moved to symcheck.pro 28/10/93

CLAUSES
/*****************************************************************/
/*    MAIN CLAUSES                                              */
/*****************************************************************/
% before each new reptype retractall all working symbol sets
   getsymbols:-
      wquit("y"),!.
   getsymbols:-
      status(1),
      clearmessage,
      makewindow(50,30,5, " SYMBOLISATION ",0,0,20,80),
      flname(kbasefile,Filename),
      consult(Filename,kbase),
      flname(symbolfile,Fname),
      consult(Fname,ksymrule),
      wcontents(Contents),          % get contents to be symbolised
      get_known(Contents,[],Rest),
      get_symbols1(Rest),
      !,
      assign_levels,
      !,
      show_symbols(Contents),
      !,
      message("end of symrule",""),pause,
      clearmessage,
      retractall(_,kbase),
      retractall(wsymbolset(_,_,_)),
      retractall(_,ksymrule),
      removewindow.

   getsymbols:-
      ermessage("failed to get symbols",""),
      retractall(_,kbase),
      retractall(_,ksymrule),
      removewindow(50,1).

   get_symbols1([]):-!.             % got symbols for all contents
   get_symbols1([Feature_class|Rest]):-
      frepresentation(Feature_class,Rep),
      !,                      % can only be one rep
                                    % cut forces backtrack to
                                    % previous feature
      symbol_rule(_,Rep,Feature_class,Features,Symbols),
      update_symbols(Feature_class,Features,Symbols),
      get_symbols1(Rest).

%   show_symbols.
% should show symbols graphically on the screen !
/*    work through list of contents
      for each feature class see if symbol for that class
      else see if sub classes & get these symbols
      need to check if point line or area
*/
   show_symbols([]):-!.
   show_symbols([Feature_class|FCs]):-
                              % single feature in class
      fsymbolism(Feature_class,Feature_class,
                  symbolspec(Type,Hue,Light,Sat,F_c,Form,Orient,Dim,Lev)),
      !,
      nl,write(Feature_class,";",Feature_class,";",Type,";",Hue,";",Light,";",
                  Sat,";",F_c,";",Form,";",Orient,";",Dim,";",Lev),
      pause,
      show_symbols(FCs).
   show_symbols([Feature_class|_]):-      % sub classes
      fsymbolism(Feature_class,Feature,
                  symbolspec(Type,Hue,Light,Sat,F_c,Form,Orient,Dim,Lev)),
      nl,write(Feature_class,";",Feature,";",Type,";",Hue,";",Light,";",
                  Sat,";",F_c,";",Form,";",Orient,";",Dim,";",Lev),
      fail.                   % get next subclass
   show_symbols([_|FCs]):-
      pause,
      !,
      show_symbols(FCs).
   show_symbols(_):-
      ermessage("failed in show_symbols","").

   update_symbols(_,_,[]):-!.       % empty list - done.
   update_symbols(Feature_class,[Feature|Rest],[Symbol|Symbols]):-
                              % normal case - assert new set of symb
      assertz(fsymbolism(Feature_class,Feature,Symbol)),!,
      update_symbols(Feature_class,Rest,Symbols).
   update_symbols(Feature_class,_,_):-    % alternative on backtrack
                              % retract previous set of symb & fail
      retractall(fsymbolism(Feature_class,_,_)),
      fail.

/**************************************************************/
/***  SYMBOLISATION RULES                              ***/
/**************************************************************/
/*** CHECK FOR SYMBOL ALREADY SPECIFIED ***/
   get_known([],In,In):-!.
   get_known([Feature_class|Rest],In,Out):-
      symbol_known(_,Feature_class,[],Fs),      % last var is list of those
                                          % unknown in a class with some
                                          % known should do something
      !,                            % symbols in class known
      nl,write_a_list(Fs),
      get_known(Rest,In,Out).
   get_known([Feature_class|Rest],In,Out):-
                              % none known in class
      append(In,[Feature_class],Work),
      get_known(Rest,Work,Out).

% only 1 in class
   symbol_known(001,Feature_class,[],[]):-
                              % class already symbolised
      message("rule 001",Feature_class),
      fsymbolism(Feature_class,Feature_class,_),
      !.

   symbol_known(002,Feature_class,[],[]):-
                              % symbol in kbase (rare)
      message("rule 002",Feature_class),
      ksymbolspec(Feature_class,Feature_class,Symbol),
      check_symbol(Feature_class,Symbol),
      !,
      assertz(fsymbolism(Feature_class,Feature_class,Symbol)).

% SUB CLASSES
/*
   symbol_known(003,Feature_class,[],[]):-
                              % sub class symbolised
                              % if one done then assume all
%nl, write("entered s_k 003"),
      get_included(Feature_class,[Feature|_]),
      fsymbolism(Feature_class,Feature,_),
      !.
*/
   symbol_known(004,Feature_class,_,Out):-
                              % sub class in kbase
                              % if one there assume all !!!
      get_included(Feature_class,[Feature|Features]),
      ksymbolspec(Feature_class,Feature,_),
      !,
      symbol_known1(Feature_class,[Feature|Features],[],Out).

   symbol_known1(_,[],Out,Out).
   symbol_known1(Feature_class,[Feature|Features],Working,Unknown):-
      ksymbolspec(Feature_class,Feature,Symbolspec),
      !,
      assertz(fsymbolism(Feature_class,Feature,Symbolspec)),
      symbol_known1(Feature_class,Features,Working,Unknown).
   symbol_known1(Feature_class,[Feature|Features],Working,Out):-
                              % symbol for feature not known or clash
      append(Working,[Feature],Newwork),
      symbol_known1(Feature_class,Features,Newwork,Out).

   symbol_rule(90,_,Feature_class,[Feature_class],[Symbol]):-
                              % look for associations with
                              % symbolised features
      kassociate_with(Feature_class,Other,symbol,_),
      fsymbolism(Other,_,Symbol),
      !.
   symbol_rule(91,_,Feature_class,[Feature_class],[Symbol]):-
                              % look for associations with
                              % symbolised features
      kassociate_with(Other,Feature_class,symbol,_),
      fsymbolism(Other,_,Symbol),
      !.
% SETTLEMENTS
   symbol_rule(105,"ranked points","Settlements",Features,Symbols):-
      get_included("Settlements",Features),
      findall(Hue,kconvention("Settlements",_,hue,Hue,_),Hues),
      assign_line_colour(Hues,[dark],[mid],
                  Hue,Lightness,Saturation),
      ksymbollist("Settlements",Shapes),
      !,
      assign_point_forms(Features,Hue,Lightness,Saturation,
                        geometric,Shapes,0,small,0,[ ],Working),
      assign_ranked_sizes(Features,[medium,small,v_small],Working,Symbols).
% ROADS
   symbol_rule(106,"network - link & node","Roads",Features,Symbols):-
      get_included("Roads",Features),
      !,
      findall(Hue,kconvention("Roads",_,hue,Hue,_),Hues),
      assign_line_colour(Hues,[mid],[mid],
                  Hue,Lightness,Saturation),
      assign_symbols(Features,
      [symbolspec(line,Hue,Lightness,Saturation,cased,main_highways,
                        0,thick,0),
                  symbolspec(line,Hue,Lightness,Saturation,cased,highways,0,
                        thick,0),
      symbolspec(line,Hue,Lightness,Saturation,cased,other_roads,0,
                        thin,0)],
                  [ ],Symbols).
% CONTOURS
   symbol_rule(107,"isolines","Relief",["c200","c500","c1000","c2000"],
                  [Symbol,Symbol,Symbol,Symbol]):-
      fbase_info("Minor Relief",_),
      !,
      ksymbolspec("Relief","contours",Symbol),
      !.
   symbol_rule(108,"isolines","Relief",["c500","c2000"],
                  [Symbol,Symbol]):-
      fbase_info("Main Relief",_),
      !,
      ksymbolspec("Relief","contours",Symbol),
      !.

   symbol_rule(109,"isolated areas","Relief",[ ],[ ]):-
      % should be able to overprint pattern, but not in BGI
      !,
      message("current system cannot overprint",
            "relief omitted from map"),pause.

/***  POINTS  ***/
```

```
   symbol_rule(01101,"dot distribution - individuals",Feature_class,
      [Feature_class],
      [symbolspec(point,Hue,Lightness,Saturation,geometric,dot,0,small,0)]):-
      ksymbollist(point_colours,Hues),
      !,
      assign_point_colour(Hues,[dark],[mid],Hue,Lightness,Saturation),
      !.

   symbol_rule(02101,"categorised points",Feature_class,Categories,Symbols):-
      klist(Feature_class,Categories),
      length_of(Categories,0,Number),
      ksymbollist(geometric_points,Shapes),
                                    % check for specific look up table
                                    % or ask user as alternatives
      length_of(Shapes,0,Num),
      Num >= Number,                % can be done with single colour
      !,
      ksymbollist(point_colours,Hues),
      !,
      assign_point_colour(Hues,[dark],[mid],Hue,Lightness,Saturation),
      !,
%      assign_point_size(_,small),  % should compute coverage
      !,
      assign_point_forms(Categories,Hue,Lightness,Saturation,
                        geometric,Shapes,0,small,0,[],Symbols).

   symbol_rule(02101,"categorised points",Feature_class,Categories,Symbols):-
                              % long list of features -
                              % needs more than 1 colour
      klist(Feature_class,Categories),
      length_of(Categories,0,Number),
      ksymbollist(geometric_points,Shapes),
                                    % check for specific look up table
                                    % or ask user as alternatives
      length_of(Shapes,0,Num),
                              % more symbols than shapes
                              % use more than 1 colour
      /* can be solved by splitting into groups matching no of shapes
      assign new colour to each group ( remove hue form list when used)
      currently not implemented */
      message("to many points categories for current implementation",
      " not symbolised"),
      pause.

   symbol_rule(03101,"ranked points",Feature_class,Features,Symbols):-
      % only instance of this is settlements
      klist(Feature_class,Allfeats),
      ask_categories(Feature_class,Allfeats,Features),
      ksymbollist(ranked_points,Shapes),
%      Hues = [red,magenta,black,yellow], % get from kbase?
      assign_point_colour([red,magenta,black,yellow],[dark],[mid],
                  Hue,Lightness,Saturation),
      !,
      assign_point_forms(Features,Hue,Lightness,Saturation,
                        geometric,Shapes,0,small,0,[],Working),
      assign_ranked_sizes(Features,[medium,small,v_small],Working,Symbols).

   symbol_rule(04101,"proportional - classed",Feature_class,
                        Classes,Symbols):-
      assign_point_colour([red,magenta,green,purple,yellow,blue,grey],
                  [mid],[high],Hue,Lightness,Saturation),
     !,
     Form_type = geometric,
     Shape = dot,
     ask_classes(Feature_class,Classes),
     assign_grad_sizes(Classes,Hue,Lightness,Saturation,Form_type,Shape,
                      0,medium,0,[],Symbols).
% could later ask relative size of symbols required rather than set at med

/***  LINES  ***/
   symbol_rule(11101,"boundaries - one level",Feature_class,[Feature_class],
         [symbolspec(line,Hue,Lightness,Saturation,continuous,"",0,
         medium,0)]):-
                           % important base info
     fbase_info(Feature_class,Priority),
     Priority >= 8,
     !,
     ksymbollist(line_colours,Hues),
     !,
     assign_line_colour(Hues,[dark],[mid],Hue,Lightness,Saturation).

   symbol_rule(11102,"boundaries - one level",Feature_class,[Feature_class],
         [symbolspec(line,Hue,Lightness,Saturation,continuous,"",0,
         medium,0)]):-
                           % less important base info
     fbase_info(Feature_class,_),
     !,
     ksymbollist(line_colours,Hues),
     !,
     assign_line_colour(Hues,[mid,dark],[low,mid],
                 Hue,Lightness,Saturation).

   symbol_rule(11103,"boundaries - one level",Feature_class,[Feature_class],
         [symbolspec(line,Hue,Lightness,Saturation,continuous,"",0,
         thick,0)]):-
                       % theme info
     ksymbollist(line_colours,Hues),
     !,
     assign_line_colour(Hues,[dark],[high],Hue,Lightness,Saturation).

   symbol_rule(11201,"boundaries - hierarchy",Feature_class,
                 [],[]):-
     message("boundaries - hierarchy : still to be programmed",
     Feature_class),
     pause,fail.

   symbol_rule(12101,"network - link & node",Feature_class,
                 [Feature_class],[]):-
         message("network - link & node",
         "still to be programmed"),
     pause, fail.

   symbol_rule(12201,"network - branching",Feature_class,
                 [Feature_class],[]):-
     message("network - branching",
         "still to be programmed"),
     pause, fail.

   symbol_rule(13101,"isolines",Feature_class,[Feature_class],[]):-
     message("isolines",
         "still to be programmed"),
     pause, fail.
```

```
/***  AREAS ***/
   symbol_rule(22101,"unclassed areas - one level",Feature_class,Groups,
            Symbols):-
      clearwindow,
      status(0),
      makewindow(54,31,5,Feature_class,3,5,15,70),
      message("   a minimum of four colours is required  ",
      "   the system is currently limited to the possibilities shown"),
      menu(5,20,5,7,["  4","  5","  6"],
                  " Select number of colours ",2,Choice),
      Num = Choice + 3,
      str_int(N,Num),
      concat("group_names",N,Name),
      kstringlist(Name,Groups),
      !,
      removewindow(54,1),
      clearmessage,
      assign_equiv_colours(Feature_class,Groups,Symbols).

   symbol_rule(22201,"unclassed areas - hierarchy",
      "Administrative areas",Features,Symbols):-
% for now assume only States to be coloured - pass to one level
      !,
      symbol_rule(_,"unclassed areas - one level","Administrative areas",
                  Features,Symbols).

   symbol_rule(22202,"unclassed areas - hierarchy",Feature_class,[],[]):-
% check what level required & pass middle or lower to 22101
      concat(Feature_class,"  unclassed areas - hierarchy",Outstring),
      message(Outstring,
            "still to be programmed"),
      pause, fail.

   symbol_rule(23101,"categorical - one level",Feature_class,
                  [Feature_class],[]):-
      message("categorical - one level",
            "still to be programmed"),
      pause, fail.

   symbol_rule(23201,"categorical - hierarchy",Feature_class,
                  [Feature_class],[]):-
      message("categorical - hierarchy",
            "still to be programmed"),
      pause, fail.

   symbol_rule(24101,"graded series - unipolar",Feature_class,
                  Classes,Symbols):-
      message("graded series - unipolar",""),
      ask_classes(Feature_class,Classes),
      kconvention("graded series - unipolar",_,hue,Hue,_),
/*    simple solution for now. should really ask user if wants
      single hue or part spectral.
      suggest single hue for < 5
      part spectral for >= 5
      findall kconventions and rank.
      */
      assign_tints(Classes,Hue,dark,high,Symbols),
      !.

   symbol_rule(24201,"graded series - bipolar",Feature_class,
                  [Feature_class],[]):-
      message("graded series - bipolar",
            "still to be programmed"),
      pause, fail.

   symbol_rule(25101,"layers - unipolar",Feature_class,
                  [Feature_class],[]):-
      message("layers - unipolar",
            "still to be programmed"),
      pause, fail.

   symbol_rule(25201,"layers - bipolar",Feature_class,
                  [Feature_class],[]):-
      message("layers - bipolar",
            "still to be programmed"),
      pause, fail.

   symbol_rule(26101,"hypsometric layers",_,
            ["0-200","200-500","500-1000","1000-2000","over 2000"],
                  Symbols):-
      fbase_info("Minor Relief",_),
      !,
      ksymbolset(minor_relief_layers,_,_,Symbols),
      !.

   symbol_rule(26102,"hypsometric layers",_,
            ["0-500","500-2000","over 2000"],Symbols):-
      fbase_info("Main Relief",_),
      !,
      ksymbolset(main_relief_layers,_,_,Symbols),
      !.

% dummy rule for testing
   symbol_rule(000,_,Feature_class,[Feature_class],
             [symbolspec(type,hue,light,sat,code,form,0,dim,0)]):-!.

/***  ASSIGNING GRAPHIC VARIABLES ***/
% all of these should include some element of asking user

   assign_point_colour([Hue1|_],[L1|_],[S1|_],Hue1,L1,S1):-
      check_colour(point,Hue1,L1,S1),
      !.                         % hue1 ok
   assign_point_colour([_|Hrest],Lrest,Srest,Hue,L,S):-
                        % only check & change hue for now
      assign_point_colour(Hrest,Lrest,Srest,Hue,L,S).

   assign_point_forms([],_,_,_,_,_,_,_,Symbols,Symbols):-!.
   assign_point_forms([_|Fs],Hue,Lightness,Saturation,
            Form_type,[S1|Shapes],Orientation,Size,Layer,
            Insymbols,Outsymbols):-
      !,
      append(Insymbols,[symbolspec(point,Hue,Lightness,Saturation,
            Form_type,S1,Orientation,Size,Layer)],Working),
      assign_point_forms(Fs,Hue,Lightness,Saturation,Form_type,Shapes,
                  Orientation,Size,Layer,Working,Outsymbols).

   assign_ranked_sizes([],_,Symbols,Symbols):-
      !.                         % all done
   assign_ranked_sizes([_|Fs],[S1|Ss],
      [symbolspec(Type,Hue,Light,Sat,Ft,F,O,_,Lay)|Insymb],
      [symbolspec(Type,Hue,Light,Sat,Ft,F,O,S1,Lay)|Outsymb]):-
   assign_ranked_sizes(Fs,Ss,Insymb,Outsymb).

assign_grad_sizes(Classes,Hue,Lightness,Saturation,Form_type,Shape,
                     0,_,Lev,[],Symbols):-
   length_of(Classes,Num),
   str_int(No,Num),
   concat("graduated",No,Key),
   ksymbollist(Key,Sizes),
   !,
   assign_grad_sizes1(Classes,Hue,Lightness,Saturation,
               Form_type,Shape,0,Sizes,Lev,[],Symbols).

assign_grad_sizes1([],_,_,_,_,_,_,_,_,Symbols,Symbols).
assign_grad_sizes1([_|Classes],Hue,Lightness,Saturation,Form_type,
         Shape,Orientation,[Size1|Sizes],Lev,Insymbols,Outsymbols):-
   append(Insymbols,[symbolspec(point,Hue,Lightness,Saturation,
               Form_type,Shape,Orientation,Size1,Lev)],Worksymbols),
   assign_grad_sizes1(Classes,Hue,Lightness,Saturation,Form_type,Shape,
                     0,Sizes,Lev,Worksymbols,Outsymbols).

assign_line_colour([Hue1|_],[L1|_],[S1|_],Hue1,L1,S1):-
   check_colour(line,Hue1,L1,S1),
   !.                        % hue1 ok
assign_line_colour([_|Hrest],Lrest,Srest,Hue,L,S):-
                     % only check & change hue for now
   assign_line_colour(Hrest,Lrest,Srest,Hue,L,S).

assign_equiv_colours(Feature_class,Features,Symbols):-
   !,
   ask_symbolset(Feature_class,equiv_area_colours,Set),
   ksymbolset(equiv_area_colours,Set,_,Symbollist),
   !,
   assign_symbols(Features,Symbollist,[],Symbols).

assign_equiv_colours(point,Features,Symbols):-
   !,
   ksymbolset(equiv_point_colours,Symbollist),
   assign_symbols(Features,Symbollist,[],Symbols).
assign_equiv_colours(line,Features,Symbols):-
   !,
   ksymbolset(equiv_line_colours,Symbollist),
   assign_symbols(Features,Symbollist,[],Symbols).

assign_tints(Classes,Hue,Light,Sat,Symbols):-
   length_of(Classes,Num),
   str_int(Chr,Num),
   concat("choro_tints",Chr,Name),
   ksymbollist(Name,Tints),!,
   assign_tints1(Classes,Hue,Light,Sat,Tints,[],Symbols).

assign_tints1([],_,_,_,_,Symbols,Symbols):-!.
assign_tints1([Class|Rest],Hue,Light,Sat,[T1|Ts],Working,Symbols):-
   append(Working,[symbolspec(Class,Hue,Light,Sat,tint,T1,0,"",0)],
               Newwork),
   !,
   assign_tints1(Rest,Hue,Light,Sat,Ts,Newwork,Symbols).

assign_symbols([],_,Symbols,Symbols):-!.     % all features done
assign_symbols(_,[],So_far,So_far):-
   !,
   ermessage("insufficient symbols in set",
                  "some features not symbolised").
   assign_symbols([_|Features],[Symbol|Symbollist],
            Insymbols,Outsymbols):-
      check_symbol(Symbol),         % if ok continue, else next symbol
      !,
      append(Insymbols,[Symbol],Working), % add symbol to list
      assign_symbols(Features,Symbollist,Working,Outsymbols).% next feature
   assign_symbols(Features,[_|Symbollist],Insymbols,Outsymbols):-
                        % after check_symbol fail - try next symbol
      assign_symbols(Features,Symbollist,Insymbols,Outsymbols).


/*** get various lists or values ***/
   get_included(Feature_class,Features):- % get base items for class
                                    % from frame
      findall(Feature,kmember_of(Feature_class,Feature),List),
      get_included1(List,[],Features).

   get_included1([],Features,Features):-!.
   get_included1([First|Rest],In,Out):-
      fbase_info_list(List),
      member(First,List),
      !,
      append(In,[First],Work),
      get_included1(Rest,Work,Out).
/*   get_included1([_|Rest],In,Out):-     % this is the general case
      get_included1(Rest,In,Out).   % it can be replaced by that below
*/                                  % as always have higher level items
   get_included1(_,In,In).          % ie stop when item not included

/*** ASK VARIOUS THINGS OF THE USER ***/
   ask_symbolset(Feature_class,Type,Set):-      % choose a set of symbols
      findall(Name,ksymbolset(Type,Name,_,_),Names),
                              %  find all sets for type
      clearwindow,
      status(0),
      clearmessage,
      makewindow(54,31,5,Feature_class,3,5,15,70),
      menu(5,20,5,7,Names,
      " Select set of symbols ",1,Choice),

      member_from_index(Names,Choice,Set),
      !,
      removewindow(54,1).
   ask_symbolset(Feature_class,_,""):-
      ermessage("failed in ask_symbolset",Feature_class),
      removewindow(54,1).


   ask_categories(Feature_class,_,Required):-   % check to see if in frame
      fcategories(Feature_class,Required),!.
   ask_categories(Feature_class,Categories,Required):-
      clearwindow,
      write(" Feature class : ",Feature_class),
      append(["All"],Categories,All),
      status(4),!,
      longmenu_mult(5,20,15,7,5,All," Which categories are requred ",[1],
                        Choices),
      list_from_index(All,Choices,Out),
      ask_categories1(Categories,Out,Required),
      assert(fcategories(Feature_class,Required)).    % add to frame
   ask_categories1(_,[],[]):-!,quit.
   ask_categories1(Categories,Out,Categories):-
      member("All",Out),!.
   ask_categories1(_,Out,Out).

   ask_classes(Feature_class,Classes):-         % check if in frame
      fcategories(Feature_class,Classes),!.
   ask_classes(Feature_class,Classes):-
%   findall types of classes for rep
      clearwindow,
      status(0),
      clearmessage,
      makewindow(54,31,5,Feature_class,3,5,15,70),
      menu(5,20,5,7,[" No classes - continuous ",
                 " Exploratory classes ",
                       " Equal intervals ",
                 " Quantile ",
                       " Nested ",
                       " User specified "],
      " Select type of class intervals ",2,Choice),
      askclasses1(Feature_class,Choice,Classes,Intervals),
      !,
      assert(fclass_intervals(Feature_class,Intervals)),
      message ("finished seting classes","next feature"),
      removewindow(54,1).
   ask_classes(Feature_class,[]):-        % check if in frame
      ermessage("failed in ask_classes",Feature_class),
      removewindow(54,1).

   askclasses1(_,0,[],[]):-!,quit.
   askclasses1(_,1,[continuous],[]):-!,
      message("continous symbolisation currently not available",
            "choose another option"),
      fail.
   askclasses1(Feature_class,2,Classes,Intervals):-   % exploratory
      !,
      clearwindow,
      status(0),
      clearmessage,
      menu(5,20,5,7,["  2","  3","  4","  5","  6","  7","  8"],
            " Select number of exploratory classes ",4,Choice),
      Num = Choice + 1,
      str_int(C,Num),
      concat("class_names",C,Name),
      kstringlist(Name,Classes),
      !,
      setclasses(exploratory,Num,Feature_class,Intervals).

   askclasses1(Feature_class,3,Classes,Intervals):-   % equal intervals
      !,
      clearwindow,
      status(0),
      clearmessage,
      menu(5,20,5,7,["  2","  3","  4","  5","  6","  7","  8"],
            " Select number of equal classes ",4,Choice),
      Num = Choice + 1,
      str_int(C,Num),
      concat("class_names",C,Name),
      kstringlist(Name,Classes),
      !,
      setclasses(equal,Num,Feature_class,Intervals).

   askclasses1(Feature_class,4,Classes,Intervals):-   % quantile
      !,
      clearwindow,
      status(0),
      clearmessage,
      menu(5,20,5,7,["  2","  3","  4","  5","  6","  7","  8"],
            " Select number of quantile classes ",4,Choice),
      Num = Choice + 1,
      str_int(C,Num),
      concat("class_names",C,Name),
      kstringlist(Name,Classes),
      !,
      setclasses(quantile,Num,Feature_class,Intervals).
   askclasses1(Feature_class,5,Classes,Intervals):-   % nested
      !,
      clearwindow,
      status(0),
      clearmessage,
      menu(5,20,5,7,["  2",
                  "  4",
                  "  8"],
         " Select number of nested classes ",2,Choice),
      str_int(C,Choice),
      concat("class_names",C,Name),
      kstringlist(Name,Classes),
      !,
      Num = Choice + 50,
      askclasses1(Feature_class,Num,_,Intervals).
   askclasses1(Feature_class,51,Classes,Intervals):-
      !,
      kstringlist("class_names2",Classes),
      !,
      setclasses(nested,2,Feature_class,Intervals).
   askclasses1(Feature_class,52,Classes,Intervals):-
      !,
      kstringlist("class_names4",Classes),
      !,
      setclasses(nested,4,Feature_class,Intervals).
   askclasses1(Feature_class,53,Classes,Intervals):-
      !,
      kstringlist("class_names8",Classes),
      !,
      setclasses(nested,8,Feature_class,Intervals).
   askclasses1(Feature_class,_,[],Intervals):-  % user set or anything else
% NEEDS TO BE WRITTEN - replace null list by user names or from kstringlist
      !,
      setclasses(user,5,Feature_class,Intervals).

   ask_bipolar(Feature_class,Classes):-
%   findall types of classes for rep
      clearwindow,
      status(0),
      clearmessage,
      makewindow(55,31,5,Feature_class,3,5,15,70),
      menu(2,20,5,7,[" No classes - continuous ",
                        " Equal intervals ",
                  " Quantile ",
                        " Nested ",
                        " Std Deviations",
                    " User specified "],
            " Select type of class intervals ",2,Choice),
    askbipolar1(Feature_class,Choice,Classes,Intervals),
    assertz(fclass_intervals(Feature_class,Intervals)),
    removewindow(55,1).

askbipolar1(_,0,_,_):-!,quit.
askbipolar1(_,1,[continuous],_):-!,
    message("continous symbolisation currently not available",
            "choose another option"),
    fail.
askbipolar1(Feature_class,2,Classes,Intervals):-   % equal intervals
    !,
    clearwindow,
    status(0),
    clearmessage,
    message("If odd number of classes, mid class will be neutral",
    " If even number of classes, mid point between central classes"),
    menu(5,20,5,7,["  2","  3","  4","  5","  6","  7","  8"],
       " Select number of equal classes ",4,Choice),
    clearmessage,
    Num = Choice + 1,
    str_int(C,Num),
    concat("class_names",C,Name),
    kstringlist(Name,Classes),
    setclasses(equal,Num,Feature_class,Intervals).
askbipolar1(Feature_class,3,Classes,Intervals):-   % quantile
    !,
    clearwindow,
    status(0),
    clearmessage,
    message("If odd number of classes, mid class will be neutral",
          " If even number of classes, mid point between central classes"),
    menu(5,20,5,7,["  2","  3","  4","  5","  6","  7","  8"],
          " Select number of quantile classes ",4,Choice),
    clearmessage,
    Num = Choice + 1,
    str_int(C,Num),
    concat("class_names",C,Name),
    kstringlist(Name,Classes),
    setclasses(quantile,Num,Feature_class,Intervals).
askbipolar1(Feature_class,4,Classes,Intervals):-   % nested
    !,
    clearwindow,
    status(0),
    clearmessage,
    message(" neutral point between central classes",""),
    menu(5,20,5,7,["  2",
                   "  4",
                   "  8"],
       " Select number of nested classes ",2,Choice),
    clearmessage,
    str_int(C,Choice),
    concat("class_names",C,Name),
    kstringlist(Name,Classes),
    Num = Choice + 40,
    askbipolar1(Feature_class,Num,_,Intervals).
askbipolar1(Feature_class,41,_,Intervals):-
    !,
    setclasses(nested,2,Feature_class,Intervals).
askbipolar1(Feature_class,42,_,Intervals):-
```

```
      !,
      setclasses(nested,4,Feature_class,Intervals).
   askbipolar1(Feature_class,43,_,Intervals):-
      !,
      setclasses(nested,8,Feature_class,Intervals).
   askbipolar1(Feature_class,_,Classes,Intervals):-% user set or anything else
% NEEDS TO BE WRITTEN - get neutral point or class, then pos & neg
      !,
      setclasses(user,5,Feature_class,Intervals),
      length_of(Intervals,Num),
      N=Num - 1,
      str_int(C,N),
      concat("class_names",C,Name),
      kstringlist(Name,Classes).

/***  SETTING CLASS INTERVALS       ***/
/*    need to read data from database
      often need to sort data
      class type  1  " No classes - continuous ",
                  2  " Exploratory classes "
                  3  " Equal intervals ",
                  4  " Quantile ",
                  5  " Nested ",
                  6  " User specified "],

*/
include "sort.pro"                    % tree sort routine

PREDICATES
   get_data(                    % read data from database
        SYMBOL,                       % feature_class
      REALLIST)                 % list of values

CLAUSES
   setclasses(exploratory,Num,Feature_class,Intervals):- % exploratory classes
      % exploratory classes use the mean of the breaks from
      % quantile and equal interval classes
      setclasses(equal,Num,Feature_class,Ints1),
      setclasses(quantile,Num,Feature_class,Ints2),
      !,
      setclasses3(Ints1,Ints2,[],Intervals).
   setclasses(equal,Num,Feature_class,Intervals):- % equal interval classes
      get_data(Feature_class,Values),
      !,
                                % get max and min values from database
      maxinlist(Values,Max),
      mininlist(Values,Min),
      Interval = (Max - Min) / Num,
      setclasses1(Min,Interval,Num,[Min],C),
      append(C,[Max],Intervals),
      retractall(wreallist(Feature_class,_)),
      retractall(_,meta).
   setclasses(quantile,Num,Feature_class,Intervals):- % for testing set all to
                                      % equal interval classes
      get_data(Feature_class,Values),
      !,                              % get max and min values from database
      maxinlist(Values,Max),
      mininlist(Values,Min),
      length_of(Values, Tot),
      No_in_class = Tot / Num,
      sort(Values,Ordered),
   setclasses2(Num,No_in_class,Ordered,[],C),  % get class breaks
   append([Min],C,C2),                  % add max & min values
   append(C2,[Max],Intervals),
   retractall(wreallist(Feature_class,_)),
   retractall(_,meta).

setclasses(_,Num,Feature_class,Intervals):- % for testing set all others
                                  % to equal interval classes
   message("chosen class type not available", "equal intervals selected"),
   pause,
   setclasses(equal,Num,Feature_class,Intervals).

setclasses1(_,_,1,In,In):-!.                  % get eq int class breaks
setclasses1(Min,Interval,Num,In,Out):-
   Next = Min + Interval,
   append(In,[Next],Working),
   Num1 = Num - 1,!,
   setclasses1(Next,Interval,Num1,Working,Out).

setclasses2(1,_,_,In,In):-!.                  % last or 1 class
setclasses2(Num,No_in_class,Ordered,In,Out):- % get quantile class breaks
   New = Num - 1,
   Pos = trunc(New * No_in_class),
   Pos1 = Pos - 1,
   member_from_index(Ordered,Pos,Val),
   member_from_index(Ordered,Pos1,Val1),
   Break = (Val + Val1) / 2,
   append([Break],In,Work),
   !,
   setclasses2(New,No_in_class,Ordered,Work,Out).

setclasses3([],_,Intervals,Intervals). % calcs means for exploratory
setclasses3([H1|Ints1],[H2|Ints2],Inlist,Intervals):-
   Break = (H1 + H2) / 2,
   append (Inlist,[Break],Worklist),
   !,
   setclasses3(Ints1,Ints2,Worklist,Intervals).

get_data(Feature_class,Values):-
   flname(metafile,Filename),
   consult(Filename,meta),
   kmeta_data(Feature_class,_,_,_,_,_,_,_,Datafile,Dataname),
   kdata_file(Datafile,_,_,_,Vars,Flname,_),
   listposition(Vars,Dataname,0,Pos),
   load_data(Flname,Pos,Values),
   !,
   retractall(_,meta),
   message("data loaded","").

load_data(Flname,Num,_):-
   assert(wreallist(work,[])),
   openread(input,Flname),
   readdevice(input),
   repeatread(input),
   readval(Num,Val),
   message("reading data",""),
   load_data1(Val),
   fail,!.
load_data(_,_,Values):-
   !,
   readdevice(keyboard),
      closefile(input),
      wreallist(work,Values),
      !,
      retractall(wreallist(work,_)).

   load_data1(Val):-
      !,
      wreallist(work,In),
      !,
      append(In,[Val],Out),
      retractall(wreallist(work,_)),
      assert(wreallist(work,Out)).

   readval(1,Val):-
      readterm(plotdata,data(Val,_,_,_,_)).
   readval(2,Val):-
      readterm(plotdata,data(_,Val,_,_,_)).*/
   readval(3,Val):-
      readterm(plotdata,data(_,_,Val,_,_)).
   readval(4,Val):-
      readterm(plotdata,data(_,_,_,Val,_)).
   readval(5,Val):-
      readterm(plotdata,data(_,_,_,_,Val)).
   readval(6,Val):-
      readterm(plotdata,node(_,_,_,_,_,Val)).

   repeatread(_).
   repeatread(File):-not(eof(File)),!,repeatread(File).

   calc_no_in_window(_,Number):-
% get data for feature class & do point in polygon to determin number
% for now settle on total number of points in dataset
      Number = 200.

/*****************************************************************/
/* END OF SYMRUL.PRO                                             */
/*****************************************************************/
```

```
/************************************************************/
/*                                                    */
/*    5B LEVELS - levels.pro                          */
/*                                                    */
/************************************************************/
/*    PROCESS
      1 - get list of symbolised features
      2 - work down list assigning levels
*/
/************************************************************/
/*                                                    */
/*    20/4/94           Intial program                */
/*                                                    */

project "mapdes"
include "globdoms.pro"
   include "tools\\tpreds.pro"
   include "tools\\status.pro"
   include "tools\\menu.pro"
   include "tools\\lineinp.pro"
   include "utils.pro"

/*  GLOBAL DATABASE - WORK  %  PREDICATES USED
%  wintegerlist(levels,sorted_list) % ORDERED LIST OF DISPLAY LEVELS
*/
/*************************************************************/
/***  SORT                                            ***/
/*** Turbo Prolog 3.3 user guide p.254 - nb error     ***/
/*** TP 3.2 UG p.492 - correct version                ***/
/*** modified by D.F. to go low > high                ***/
/*** note - also eliminates duplicate values          ***/

DOMAINS
%  INTEGERLIST = INTEGER*
  INTTREE = reference t(VAL,INTTREE,INTTREE)
  VAL = INTEGER

PREDICATES
   sort(INTEGERLIST,           % inlist
      INTEGERLIST)             % sorted list
   insert(INTEGER,INTTREE)
   instree(INTEGERLIST,INTTREE)
   treemembers(INTEGER,INTTREE)

CLAUSES
   insert(Val,t(Val,_,_)):-!.
   insert(Val,t(Val1,Tree,_)):- Val>Val1,!,insert(Val,Tree).
   insert(Val,t(_,_,Tree)):- insert(Val,Tree).

   instree([],_).
   instree([H|T],Tree):-
      insert(H,Tree),
      instree(T,Tree).

   treemembers(_,T) :-         free(T),!,fail.
   treemembers(X,t(_,_,R)) :-        treemembers(X,R).
   treemembers(X,t(Refstr,_,_)) :- X=Refstr.
   treemembers(X,t(_,L,_)) :-        treemembers(X,L).

   sort(Lin,Lout):-
      instree(Lin,Tree),
      findall(X,treemembers(X,Tree),Lout).

/***  SET LEVELS ***/
PREDICATES
%  assign_levels              declared in globdoms
   set_levels
   set_levels1(SYMBOL)
   update_level(
            SYMBOL,
            SYMBOL,
            INTEGER)
   sort_levels

CLAUSES
   set_levels :-
      message("processing levels - please wait",""),
      fsymbolism(_,Feat,symbolspec(_,_,_,_,_,_,_,_,0)),
      set_levels1(Feat),
      fail.
   set_levels:-
      !,clearmessage.

   set_levels1(Feat):-!,
      fsymbolism(Feat_class,Feat,_),!,
      frepresentation(Feat_class,Rep),!,
      klevel(Feat_class,Rep,Level),!,
      update_level(Feat_class,Feat,Level).

   update_level(Feat_class,Feat,Level):-
      retract(fsymbolism(Feat_class,Feat,symbolspec(A,B,C,D,E,F,G,H,_))),!,
      assertz(fsymbolism(Feat_class,Feat,symbolspec(A,B,C,D,E,F,G,H,Level))).

   sort_levels:-
      findall(Level,
         fsymbolism(_,_,symbolspec(_,_,_,_,_,_,_,_,Level)),Levels),
      !,
      sort(Levels,Levellist),
      !,
      assert(wintegerlist(levels,Levellist)).
    sort_levels:-
    ermessage("failed in sort_levels","").

   assign_levels:-
      set_levels,!,
      sort_levels.
   assign_levels:-
      ermessage("failure in levels","").

/***  END OF LEVELS     ***/
```

```
/****************************************************************/
/*          6  DISPLAY                                  */
/****************************************************************/
/*
      D.F. October 1993 main package
*/
/*    know both representation type and symbol for each class and
       levels sorted into plotting order
      PROCESS
       -    draw features in order
            some (few) special rules for reps (? and features)
            mostly general rules for type
*/
project "mapdes"
include "globdoms.pro"
% call graphics routines and constants
   include "BGIconst.pre"           % custom version moved to \map
%   include "tools\\graph.pre"      % moved to globdoms
   include "tools\\tpreds.pro"
   include "tools\\graph.pro"
   include "tools\\status.pro"
   include "tools\\menu.pro"
   include "tools\\lineinp.pro"
   include "utils.pro"

CONSTANTS
   bgi_path = "..\\bgi"
   display_window = 60              % text window for display module
   plot_window          = 61        % main graphics window for map
                                    % includes title & border
   key_window           = 62        % window for legend, etc
   gstatus_win          = 63        % window for graphics status line
   gmess_win            = 64        % window for graphics messages
   map_window           = 65        % window for plotting map
                                    % no border or title
DATABASE - DISPLAYMOD
   wprojection(SYMBOL)
   wscale_factor(REAL)

DATABASE - LOOKUP
   kcolour_table(        % conversiopn from descriptive to BGI colours
                         % for area fill
            SYMBOL,                 % symbolic name for colour
            SYMBOL,                 % hue
            SYMBOL,                 % lightness
            SYMBOL,                 % saturation ?? brightness?
            INTEGER,                % colour number
            INTEGER,                % fill style
            SYMBOL).                % fill pattern - name
   kpattern (            % look up table for pattern definitions
            SYMBOL,                 % pattern name
            INTEGERLIST)            % 8 element list describing pattern
   kline_colour(         % conversion from descriptive to
                         % BGI line colours
            SYMBOL,                 % symbolic name for colour
            SYMBOL,                 % hue
            SYMBOL,                 % lightness
            SYMBOL,                 % saturation ?? brightness?
            INTEGER).               % colour number

   kline_form(                % conversion from descriptive to BGI val
            SYMBOL,                 % form code
            SYMBOL,                 % form
            INTEGER,                % form value
            INTEGER)                % form style if value = 4 (user)
   kline_width(               % conversion from descriptive to BGI value
            SYMBOL,                 % width code
            INTEGER)                % value
   kpoint_form(               % conversion from descriptive to BGI
            SYMBOL,                 % form code
            SYMBOL,                 % form
            INTEGER)                % BGI form value
   kpoint_size(               % conversion from descriptive to BGI value
            SYMBOL,                 % size code
            INTEGER)                % BGI value (rad in pixels)
   klookup(             % general purpose lookup values
            SYMBOL,                 % feature / name
            SYMBOL,                 % input 1
            SYMBOL,                 % input 2
            SYMBOL,                 % output 1
            SYMBOL)                 % output 2

CLAUSES     % for global graphics predicates
   maxviewport:-
      getmaxx(MaxX),
      getmaxy(MaxY),
      setviewport(0,0,MaxX,MaxY,clip).

/***  Basic device handling   ***/
PREDICATES
   setgraphics          % sets up graphics device
   closegraphics        % closes graphics device
   gstatus              % status line in graphics
            (STRING)          % message to display
   gpause               % pause until keypressed
   gf10                       % pause until F10 pressed
   gmessage (STRING)    % writes a message above status line

CLAUSES
   setgraphics:-
      message("initialising graphics",""),
      initgraph(vga,vgahi,_,_,bgi_path),
      settextstyle(default_font,horiz_dir,1).

   closegraphics:-
      closegraph(),
      makewindow(display_window,30,6," DISPLAY ",0,0,20,80),
      clearmessage,
      status(1).

   gstatus(Message):-
      shift_gwindow(Last),
      shift_gwindow(gstatus_win),
      clear_gwindow,
      getcolor(Current),
      setcolor(1),
      settextjustify(1,1),
      outtextxy(320,6,Message),
      setcolor(Current),
      shift_gwindow(Last).

   gmessage(Message):-
```

```
      shift_gwindow(Last),
      shift_gwindow(gmess_win),
      clear_gwindow,
      getcolor(Current),
      setcolor(1),
      settextjustify(1,1),
      outtextxy(320,6,Message),
      setcolor(Current),
      shift_gwindow(Last).

   gpause:-
      gstatus(" Press any key to continue "),
      readdevice(Old),
      readdevice(keyboard),
      repeat,
      readkey(_),!,
      readdevice(Old).

   gf10:-
      gstatus(" Press F10 to continue "),
      readdevice(Old),
      readdevice(keyboard),
      repeat,
      readkey(fkey(10)),!,
      readdevice(Old).

/************  GRAPHICS SCALING TEST  ****************/
PREDICATES
   world_to_virtual(          % converts from projection coordinates to
                        % integer values with origin in top left.
         REAL,                % Long
         REAL,                % Lat
         INTEGER,             % X
         INTEGER)             % Y_
   set_windows          % sets map window to appropriate size
                        % depending on shape of area, & sets scale
                        % for plotting
   check_ratio(               % checks width/height ratio & opens window
%        INTEGER,             % top
%        INTEGER,             % left
%        INTEGER,             % bottom
%        INTEGER,             % right
         REAL)                % ratio

CLAUSES
/*
   world_to_virtual(Long,Lat,X,Y_):-            % first clause
% find projection and transform accordingly - simple transform for now
% assumes all lat & long E & N - ie possitive
% input in units defined by meta data
% output in integer plot coords at scale to suit plotting
      wprojection(sphere),
      wscale_factor(Factor),
      flat_long(Minlong,_,_,Maxlat),!,
      XX = Long + 0.0001 - Minlong,
      X = round(XX * Factor),
      YY = Maxlat + 0.0001 - Lat,
      Y_ = round(YY * Factor).


   world_to_virtual(Long,Lat,X,Y_):-
```

```
% default if nothing set
% used for flimits
% find projection and transform accordingly - simple transform for now
% assumes all lat & long E & N - ie possitive
% input in dd.dd, ie degrees and decimal parts
% output dddd ie integer plot coords in hundredths of degrees
      flat_long(Minlong,_,_,Maxlat),!,
      XX = Long + 0.0001 - Minlong,
      X = round(XX * 100),
      YY = Maxlat + 0.0001 - Lat,
      Y_ = round(YY * 100).
*/
   world_to_virtual(Long,Lat,X,Y):-
      X = round(Long * 100),
      Y = round(Lat * 100).

   set_windows:-
      fmap_title(Title),
      setpalette(15,0),            % make colour 15 black
      setpalette(0,63),            % make background colour (0) white
      setpalette(4,36),            % improve red
      setpalette(12,38),           % improve orange
      setbkcolor(63),
      make_gwindow(plot_window,113,8,Title,0,0,455,482),
      make_gwindow(gmess_win,113,0,"",456,0,12,640),
      make_gwindow(gstatus_win,113,0,"",468,0,12,640),
      gstatus(" processing layout - please wait "),
      flat_long(Minlong,Maxlong,Minlat,Maxlat),
      world_to_virtual(Minlong,Minlat,VLeft,VBottom),
      world_to_virtual(Maxlong,Maxlat,VRight,VTop),
      VRatio = (VRight - VLeft) / (VTop - VBottom),
      check_ratio(VRatio),!,
gmessage ("windows set"),
      make_scale(map_window,VLeft,VRight,VBottom,Vtop).

   check_ratio(Ratio):-
      Ratio > 1.085,!,
      window_(plot_window,_,L,_,R,_,_),!,
      W = R - L,
      Height = round(W / Ratio),
      make_gwindow(map_window,8,0,"",12,1,Height,480).
   check_ratio(Ratio):-
      Ratio <= 1.085,
      window_(plot_window,_,_,_,_,B,Bot),!,
      H = B - Bot,
      Width = round(H * Ratio),
      make_gwindow(map_window,8,0,"",12,1,442,Width).

/****************************************************************/
/*                                                              */
/*    MAP PLOTTING SECTION                                      */
/*                                                              */
/****************************************************************/
include "bgiplot.pro"        % actual plotting routines
                             % for each rep type
PREDICATES
   draw_map
CLAUSES
   draw_map:-
      wintegerlist(levels,Levels),
      file_exist("kmeta.kba"),
      retractall(_,kbase),
      retractall(_,ksymrule),
      retractall(_,klocate),
      retractall(_,kreps),
      retractall(_,meta),
      consult("kmeta.kba",meta),
      consult("lookup.kba",lookup),
      plot_level(Levels),
      !,
      retractall(_,lookup),
      retractall(_,meta).
   draw_map:-
      retractall(_,lookup),
      retractall(_,meta),
      closefile(input),
      closefile(datafile),
      gmessage("failure in draw_map"),
      gpause.

/*****************************************************************/
/*                                                              */
/*    MAIN SECTION                                              */
/*                                                              */
/*****************************************************************/
PREDICATES
%   screenplot    now in globdoms

CLAUSES
   screenplot:-
      wquit("y"),!.
   screenplot:-
      status(1),
      clearmessage,
      makewindow(60,30,6," DISPLAY ",0,0,20,80),
      setgraphics,
      set_windows,!,
      draw_map,!,
      gmessage("map plotting finished"),
      gf10,
      !,
      closegraphics,
      retractall(_,displaymod),
      status(1),
      clearmessage,
      message("graphics completed",""),pause.
   screenplot:-
      closegraphics,
      retractall(_,displaymod),
      status(1),
      clearmessage,
      makewindow(60,30,6," DISPLAY ",0,0,20,80),
      message("failure in graphics",""),pause.

/***  END OF DISPLAY    ***/
```

```
/*******************************************************************/
/*** BGIPLOT.PRO                                                ***/
/*******************************************************************/
% include file for displaying map data on screen
/*
   plot_level ([Levels])      main routine - works on ordered list
      of levels - proper tail recursive routine
      for each level:-
            get feature class & rep
            get meta data
            get database file name(s)
            set up scaling
            call plotting routine for feature class &/or rep type
            clear working values

   plot_class(Feature_class,Rep_type)
      for each feature in class:
            get symbolisation
            translate to BGI settings
            assert values in database
            fail (go to next)
      for each entity in database
            set BGI settings for feature
            plot entity
            fail (go to next)
*/
/*    an alternative method for plotting complex symbols would be to have
      two (or more) sets of plotting parameters in database and to fail
      back into subsequent parts - ie order of assertion of wsetup important.
      wsetup could have less terms in this case.
*/

%DOMAINS                moved to globdoms
%   file = input

/*
DATABASE - PLOTDATA           moved to globdoms 19/9/94
   node(integer,symbol,integer,integer,symbol,real)
   chain(integer,symbol,symbol,symbol,integer,integer,integer,integerlist)
   polygon(integer,symbol,integer,real,real,real,integerlist)
*/

DATABASE - CURRENTLEVEL
% working database for current level
   wcurrent_class(SYMBOL)     % current feature class
   wcoord_file(SYMBOL)        % symbolic name of coordinate file
   wdata_file(SYMBOL)         % symbolic name of data file
   wdata_name(SYMBOL)         % name of feature class in data file
   wlook_up(                  % relate kbase name to dbase name
            SYMBOL,                 % name in kbase
            SYMBOL)                 % name in dbase
   wdata_position(INTEGER)    % position of feature class in data file
   wsetup(              % parameters for plotting feature
            SYMBOL,           % feature
            INTEGER,          % main draw colour
            INTEGER,          % line or point style
            INTEGER,          % line pattern
            INTEGER,          % line width or point size
            INTEGER,          % fill colour or second colour
            INTEGER,          % fill style or second style
            INTEGERLIST,            % fill pattern
```

```
         INTEGER)                 % second width or size
   wcomplex (
         SYMBOL,                  % feature
         INTEGER)    % 1 or 0 - flag for complex (2 part) symbols

PREDICATES
   plot_level(              % main processing pred - recursive
         INTEGERLIST)             % sorted list of levels to plot
   check_class(             % checks feature class & rep type
                            % for correct meta data
/*    there are few occasions where this arises - that more than one
      set of meta data is required - the most obvious eg is relief where
      both line & poly data are possible.
      note: plotting points from area data doesn't need change.
*/
         SYMBOL,                  % F_class input
         SYMBOL)                  % returned F_class
   check_file(              % checks to see if data exists
         SYMBOL,                  % key to file type
                                  % meta
                                  % coord
                                  % data
                                  % look_up
         SYMBOL)                  % f_class
   set_work_lookup(         % asserts lookup names of features
         SYMBOLLIST)        % in WORK - recursive
   get_position(      % gets postion of variable in datafile
         INTEGER)           % column in database
   plot_class(              % processes each feature class
         SYMBOL,                  % feature_class
         SYMBOL)                  % representation type
   plot_zones     (         % controls access to area files
         SYMBOL)                  % type of feature)
   plot_zone      (         % reads and plots a zone
         SYMBOL)                  % type of feature
   plot_lines     (         % controls access to line files
         SYMBOL)                  % Feature_class
   plot_line                % reads & plots a line
   plot_line1(              % plots background or foreground of line
         INTEGER,           % 1 = backgroung, 2 = foreground
         INTEGER,           % LC,
         INTEGER,           % LS,
         INTEGER,                 % LP,
         INTEGER,           % LW,
         INTEGERLIST)             % Coords),

   plot_points    (         % controls access to point files
         SYMBOL)                  % type of feature
   plot_point   (INTEGER)   % data position & type
   set_symbol(              % converts from descriptive values to BGI vals
                      % asserted into wsetup
         SYMBOL,                  % type - point, line, area
         SYMBOL,                  % Feature
         SYMBOL,                  % Hue
         SYMBOL,                  % lightness
         SYMBOL,                  % saturation
         SYMBOL,                  % form
         SYMBOL,                  % form code
         INTEGER,                 % orientation
         SYMBOL)                  % dimension
   setfill  (               % sets fill style or pattern for areas
            INTEGER,                % colour
            INTEGER,                % style - bgi standard patterns
            INTEGERLIST)            % user patterns - style = 12
   draw_chain     (           % recursively calls draw_line for complex line
            INTEGERLIST)            % coords
/*** non determ predicate ***/
   nondeterm repeatread(      % repeat loop until end of file
            FILE)
   readval(                   % redirects input to datafile & gets value
            INTEGER,                % position of Var in file
            REAL)                   % value of variable
   feat_look_up(              % converts from database feature code
                              % to feature name in knowledge base
            SYMBOL,                 % database feature code
            SYMBOL)                 % knowledgebase feature name
   value_to_class(            % converts data value to class for symbol
            REAL,             % i value
            SYMBOL)                 % o class
   value_to_class1(
            REAL,             % i value
            REALLIST,         % i interval list
            SYMBOLLIST,       % i class names
            SYMBOL)                 % o class

CLAUSES
   plot_level([]):- !.              % end of list
   plot_level([Level|Rest]):-       % main version of clause
      fsymbolism(F_class,_,symbolspec(_,_,_,_,_,_,_,_,Level)),
                                    % should only be one class / level
      gmessage(F_class),gpause,
      check_file(meta,F_class),
      check_file(coord,F_class),
      check_file(data,F_class),
      check_file(look_up,F_class),
      /***
         should also get projection and scale factor and set up transformation
         this is omitted for now as all data conforms to test format
      */
      assert(wcurrent_class(F_class)),
      frepresentation(F_class,Rep),
      !,
      plot_class(F_class,Rep),
      !,
      retractall(_,currentlevel),
      retractall(klook_up(_,_,_)),
      closefile(input),             % should already be closed
      closefile(datafile),          % but confirm
      !,
      plot_level(Rest).
   plot_level([Level|Rest]):-             % clause to catch failures
      !,
      fsymbolism(F_class,_,symbolspec(_,_,_,_,_,_,_,_,Level)),
      retractall(_,currentlevel),
      closefile(input),             % may not be closed due to failure
      closefile(datafile),          % so confirm
      concat(" failed to find data for ", F_class, Message),
      gmessage(Message),
      !,
      plot_level(Rest).

   check_class("Relief","Contours"):-
```

```
                        % if F_class is relief & rep type isolines
                        % then change class to contours
   % in future this could be handled by lookup in meta file
   fsymbolism("Relief",_,symbolspec(line,_,_,_,_,_,_,_,_)),
   !.
check_class(F_class,F_class).    % else continue

check_file(meta,F_class):-
   check_class(F_class,Class),
   kmeta_data(Class,_,_,_,_,_,_,_,_,_,_),
   !.
check_file(meta,F_class):-
   !,
   concat("no meta data for ",F_class,Mess),
   gmessage(Mess),gpause,
   fail.
check_file(coord,F_class):-
   check_class(F_class,Class),
   kmeta_data(Class,_,_,_,_,_,_,Coordname,_,_,_),
   kcoord_file(Coordname,_,_,_,_,_,_,_,_,Coordfile,_),
   file_exist(Coordfile),
   !,
   assert(wcoord_file(Coordfile)).
check_file(coord,F_class):-
   !,
   concat("coordinate file not found for ",F_class,Mess),
   gmessage(Mess),gpause,
   fail.
check_file(data,F_class):-       % no separate data
   check_class(F_class,Class),
   kmeta_data(Class,_,_,_,_,_,_,_,_,"",Dataname),
   !,
   assert(wdata_name(Dataname)).
check_file(data,F_class):-       % no separate data
   check_class(F_class,Class),
   kmeta_data(Class,_,_,_,_,_,_,Coordname,_,_,Dataname),
   kmeta_data(Class,_,_,_,_,_,_,_,_,Coordname,Dataname),
   !,
   kcoord_file(Coordname,_,_,_,_,_,_,_,_,Coordfile,_),
   !,
   file_exist(Coordfile),
   assert(wcoord_file(Coordfile)),
   assert(wdata_file(Coordfile)),
   assert(wdata_name(Dataname)).
check_file(look_up,F_class):-    % no look up table
   check_class(F_class,Class),
   kmeta_data(Class,_,_,_,_,_,_,_,"",_,_),
   !.
check_file(look_up,F_class):-    % look up table exists
   check_class(F_class,Class),
   kmeta_data(Class,_,_,_,_,_,_,_,L_file,_,_),
   file_exist(L_file),
   !,
   consult(L_file,meta),
   findall(Feature,fsymbolism(F_class,Feature,_),Features),
   set_work_lookup(Features),
   retractall(klook_up(_,_,_)).
check_file(look_up,F_class):-
   !,
   concat("look_up file not found for ",F_class,Mess),
   gmessage(Mess),gpause,
      fail.
   check_file(data,F_class):-
      check_class(F_class,Class),
      kmeta_data(Class,_,_,_,_,_,_,_,Datafile,Dataname),
      kdata_file(Datafile,_,_,_,Filename,_),
      file_exist(Filename),
      !,
      assert(wdata_file(Filename)),
      assert(wdata_name(Dataname)),
      get_position(Position),
      assert(wdata_position(Position)).
   check_file(data,F_class):-
      !,
      concat("data file not found for ",F_class,Mess),
      gmessage(Mess),gpause,
      fail.

   set_work_lookup([]):-!.
   set_work_lookup([Feature|Rest]):-
      klook_up(_,Feature,Code),
      !,
      assert(wlook_up(Feature,Code)),
      set_work_lookup(Rest).

   get_position(0):-
      wdata_name(""),!.
   get_position(Pos):-
      wdata_file(Datafilename),
%gmessage(Datafilename),gpause,
      kdata_file(_,_,_,_,Vars,Datafilename,_),
      wdata_name(Name),
      !,
      listposition(Vars,Name,0,Pos).

/*****************************************/
/*** PLOTTING OF REPRESENTATION TYPES ***/
/*****************************************/
/*** POINT CLASSES ***/
   plot_class(_,"dot distribution - individuals"):-
      !,
      gmessage("plotting for dot distribution - individuals not written yet"),
      gpause.
   plot_class(_,"dot distribution - groups"):-
      !,
      gmessage("plotting for dot distribution - groups not written yet"),
      gpause.
   plot_class(_,"categorised points"):-
      !,
      gmessage("plotting for categorised points not written yet"),
      gpause.
   plot_class(F_class,"ranked points"):-
      fsymbolism(F_class,Feature,symbolspec(_,H,L,S,FC,F,O,D,_)),
      set_symbol(point,Feature,H,L,S,FC,F,O,D),
      fail,!.                           % set symbol for other features
   plot_class(_,"ranked points"):-
      plot_points(f_codes),!.

   plot_class(F_class,"proportional - classed"):-
      fsymbolism(F_class,Feature,symbolspec(_,H,L,S,FC,F,O,D,_)),
      set_symbol(point,Feature,H,L,S,FC,F,O,D),
      fail,!.                           % set symbol for other features
   plot_class(_,"proportional - classed"):-
      !,
      plot_points(values),!.
   plot_class(_,"proportional - graduated"):-
      !,
%     plot larger before smaller
      gmessage("plotting for proportional - graduated not written yet"),
      gpause.
   plot_class(_,"proportional - bi-polar"):-
      !,
%     plot larger before smaller
      gmessage("plotting for proportional - bi-polar not written yet"),
      gpause.
   plot_class(_,"multivariable - fixed size"):-
      !,
      gmessage("plotting for multivariable - fixed size not written yet"),
      gpause.
   plot_class(_,"multivariable - classed"):-
      !,
%     plot larger before smaller
      gmessage("plotting for multivariable - classed not written yet"),
      gpause.

   plot_class(_,"multivariable - graduated"):-
      !,
%     plot larger before smaller
      gmessage("plotting for multivariable - graduated not written yet"),
      gpause.
   plot_class(_,"spot values"):-
      !,
      gmessage("plotting for spot values not written yet"),
      gpause.
   plot_class(_,"irregular value network"):-
      !,
      gmessage("plotting for irregular value network not written yet"),
      gpause.
   plot_class(_,"grid surface values"):-
      !,
      gmessage("plotting for grid surface values not written yet"),
      gpause.

/*** LINE CLASSES ***/
   plot_class(F_class,"boundaries - one level"):-
      fsymbolism(F_class,Feature,symbolspec(_,H,L,S,FC,F,O,D,_)),
      !,
      set_symbol(line,Feature,H,L,S,FC,F,O,D),
      !,
      plot_lines(F_class).
   plot_class(F_class,"boundaries - hierarchy"):-
      fsymbolism(F_class,Feature,symbolspec(_,H,L,S,FC,F,O,D,_)),
      set_symbol(line,Feature,H,L,S,FC,F,O,D),
      fail,!.                           % set symbol for other features
   plot_class(F_class,"boundaries - hierarchy"):-
      plot_lines(F_class),!.

   plot_class(F_class,"network - link & node"):-
      fsymbolism(F_class,Feature,symbolspec(_,H,L,S,FC,F,O,D,_)),
      set_symbol(line,Feature,H,L,S,FC,F,O,D),
      fail,!.                           % set symbol for other features
   plot_class(F_class,"network - link & node"):-
      plot_lines(F_class),!.
   plot_class(F_class,"network - branching"):-
      fsymbolism(F_class,Feature,symbolspec(_,H,L,S,FC,F,O,D,_)),
      set_symbol(line,Feature,H,L,S,FC,F,O,D),
      fail,!.                             % set symbol for other features
   plot_class(F_class,"network - branching"):-
      plot_lines(F_class),!.
   plot_class(F_class,"isolines"):-
      fsymbolism(F_class,Feature,symbolspec(_,H,L,S,FC,F,O,D,_)),
      set_symbol(line,Feature,H,L,S,FC,F,O,D),
      plot_lines(F_class),!.
   plot_class(_,"flowlines"):-
      !,
      gmessage("plotting for flowlines not written yet"),
      gpause.
   plot_class(_,"misc. tangible lines"):-
      !,
      gmessage("plotting for misc. tangible lines not written yet"),
      gpause.

/*** AREA CLASSES ***/
   plot_class(F_class,"isolated areas"):-
      fsymbolism(F_class,Feature,symbolspec(_,H,L,S,FC,F,O,D,_)),
      !,
      set_symbol(area,Feature,H,L,S,FC,F,O,D),
      !,
      plot_zones(f_code).
   plot_class(F_class,"unclassed areas - one level"):-
      fsymbolism(F_class,Feature,symbolspec(_,H,L,S,FC,F,O,D,_)),
      set_symbol(area,Feature,H,L,S,FC,F,O,D),
      fail,!.
   plot_class(F_class,"unclassed areas - one level"):-
     sort_adjacent_classes(F_class),
     plot_zones(work),
     !,
     retractall(wcolour(_,_)).
   plot_class(F_class,"unclassed areas - hierarchy"):-
      !,
      plot_class(F_class,"unclassed areas - one level").
   plot_class(_,"categorical - one level"):-
      !,
      gmessage("plotting for categorical - one level not written yet"),
      gpause.
   plot_class(_,"categorical - hierarchy"):-
      !,
      gmessage("plotting for categorical - hierarchy not written yet"),
      gpause.
   plot_class(F_class,"graded series - unipolar"):-
      fsymbolism(F_class,Feature,symbolspec(_,H,L,S,FC,F,O,D,_)),
      set_symbol(area,Feature,H,L,S,FC,F,O,D),
      fail,!.
   plot_class(_,"graded series - unipolar"):-
     plot_zones(values),!.
   plot_class(_,"graded series - bipolar"):-
      !,
      gmessage("plotting for graded series - bipolar not written yet"),
      gpause.
   plot_class(_,"graded series - bivariate"):-
      !,
      gmessage("plotting for graded series - bivariate not written yet"),
      gpause.
   plot_class(F_class,"layers - uniploar"):-
```

```
      fsymbolism(F_class,Feature,symbolspec(_,H,L,S,FC,F,O,D,_)),
      set_symbol(area,Feature,H,L,S,FC,F,O,D),
      fail,!.//
   plot_class(_,"layers - uniploar"):-
     plot_zones(values),!.
   plot_class(_,"layers - bipolar"):-
      !,
      gmessage("plotting for layers - bipolar not written yet"),
      gpause.
   plot_class(F_class,"hypsometric layers"):-
      fsymbolism(F_class,Feature,symbolspec(_,H,L,S,FC,F,O,D,_)),
      set_symbol(area,Feature,H,L,S,FC,F,O,D),
      fail,!.
   plot_class(_,"hypsometric layers"):-
     plot_zones(f_code),!.

/*** SET UP SYMBOLS FOR PLOTTING  **/
   set_symbol(area,Feature,H,L,S,solid,_,_,_):-
      kcolour_table(_,H,L,S,Col_no,Fill_style,Fill_pattern),
      !,
      kpattern(Fill_pattern,FP),!,
      assert(wsetup(Feature,Col_no,solid_line,0,norm_width,
                  Col_no,Fill_style,FP,0)).
   set_symbol(area,Feature,H,_,_,tint,Percent,_,_):-
      kline_colour(_,H,_,_,Col_no),
      kpattern(Percent,Fill_pattern),
      !,
      assert(wsetup(Feature,Col_no,solid_line,0,norm_width,
                  Col_no,12,Fill_pattern,0)).
   set_symbol(line,Feature,H,_,_,cased,FC,_,D):-
      klookup(cased,FC,H,H1,H2),
      kline_colour(_,H1,_,_,Col_for),
      kline_colour(_,H2,_,_,Col_bk),
%      kline_form(cased,FC,Fval,_), % should use lookup for form
      kline_form(cased,_,Fval,_),    % default to solid for now
      kline_width(D,Dval),
      Dval2 = 1,              % should compute but use fine line
      !,
      assert(wsetup(Feature,Col_for,Fval,0,Dval2,Col_bk,0,[],Dval)).
   set_symbol(line,_,_,_,_,cased,FC,_,_):-
      concat("unable to find lookup value for cased line - ",
                  FC, Mess),
      !,
      gmessage(Mess),gpause.
   set_symbol(line,Feature,H,L,_,complex,FC,_,D):-
      kline_colour(_,H,L,_,Col_no),
      kline_form(complex,FC,Fval,_),
      kline_width(D,Dval),
      !,
      assert(wsetup(Feature,Col_no,Fval,0,Dval,0,0,[],0)).
   set_symbol(line,Feature,H,L,_,FC,_,_,D):-
      kline_colour(_,H,L,_,Col_no),
      kline_form(FC,_,Fval,_),
      kline_width(D,Dval),
      !,
      assert(wsetup(Feature,Col_no,Fval,0,Dval,0,0,[],0)).
   set_symbol(point,Feature,H,L,_,geometric,F,_,D):-
      kline_colour(_,H,L,_,Col_no),
      kpoint_form(geometric,F,Val),
      kpoint_size(D,Size),
      !,
      assert(wsetup(Feature,Col_no,Val,0,Size,0,0,[],0)).
  set_symbol(_,Feature,_,_,_,_,_,_):-
      concat("cannot set symbol for ",Feature,Message),
      gmessage(Message),gpause,
      assert(wsetup(Feature,7,0,0,norm_width,0,0,[],0)).

/*** PLOT AREAS ***/
    plot_zones(Type):-
      wcoord_file(Filename),
      readdevice(Old),
      openread(input,Filename),
      readdevice(input),
      plot_zone(Type),
      !,
      readdevice(Old),
      closefile(input),
      closefile(datafile).
   plot_zone(f_code):-
      repeatread(input),
      readterm(plotdata,polygon(_,Fcode,_,_,_,_,Coords)),
      feat_look_up(Fcode,Feature),
      wsetup(Feature,LC,LS,LP,LW,FC,FS,FP,_),
      setcolor(LC),
      setlinestyle(LS,LP,LW),
      setfill(FC,FS,FP),
      draw_poly(Coords),
      fail.
   plot_zone(values):-
      wdata_file(Datafilename),
      openread(datafile,Datafilename),
      wdata_position(Pos),
      repeatread(input),
      readterm(plotdata,polygon(_,_,_,_,_,_,Coords)),
      readval(Pos,Val),
      value_to_class(Val,Feature),
      wsetup(Feature,LC,LS,LP,LW,FC,FS,FP,_),
      setcolor(LC),
      setlinestyle(LS,LP,LW),
      setfill(FC,FS,FP),
      draw_poly(Coords),
      fail.
   plot_zone(work):-         % uses values asserted into database WORK
      repeatread(input),
      readterm(plotdata,polygon(_,Name,_,_,_,_,Coords)),
      wcolour(Name,Group),
      wsetup(Group,LC,LS,LP,LW,FC,FS,FP,_),
      setcolor(LC),
      setlinestyle(LS,LP,LW),
      setfill(FC,FS,FP),
      draw_poly(Coords),
      fail.
   plot_zone(_).

   setfill(FC,12,FP):-             % 12 = user pattern
      !,
      setfillpattern(FP,FC).
   setfill(FC,FS,_):-
      !,
      setfillstyle(FS,FC).

/*** PLOT LINES ***/
```

```
   plot_lines(_):-
      wcoord_file(Filename),
      !,
      readdevice(Old),
      openread(input,Filename),
      readdevice(input),
      plot_line,
      !,
      readdevice(Old),
      closefile(input).
   plot_line:-
      repeatread(input),
      readterm(plotdata,chain(_,Feature,_,_,_,_,_,Coords)),
      wsetup(Feature,LC,LS,LP,LW,C2,S2,_,W2),
      plot_line1(1,C2,S2,$FFFF,W2,Coords),     % plot background
      plot_line1(2,LC,LS,LP,LW,Coords),   % plot foreground
      fail.
   plot_line.

   plot_line1(1,0,0,_,0,_):-!.                  % no background
   plot_line1(1,C2,S2,P2,W2,Coords):-           % background part
      !,
      setcolor(C2),
      setlinestyle(S2,P2,W2),
      draw_chain(Coords).
   plot_line1(2,LC,0,LP,LW,Coords):-            % solid line - normal
      !,
      setcolor(LC),
      setlinestyle(0,LP,LW),
      draw_chain(Coords).
   plot_line1(2,LC,LS,LP,LW,Coords):-     % patterned line - clear
                                          % background first
      setcolor(LC),
      setlinestyle(LS,LP,LW),
      draw_chain(Coords).

   draw_chain([_|[_|[]]]):-!.
   draw_chain([X1|[Y1|[X2|[Y2|Rest]]]]):-
      draw_line(X1,Y1,X2,Y2),!,
      draw_chain([X2,Y2|Rest]).

/*** PLOT POINTS ***/
   plot_points(f_codes):-           % coords & feature codes in same file
      wcoord_file(Filename),
      !,
      readdevice(Old),
      openread(input,Filename),
      readdevice(input),
      plot_point(-2),
      !,
      readdevice(Old),
      closefile(input).
   plot_points(values):-            % coords & data in same file
      wcoord_file(Filename),
      wdata_file(Filename),
      !,
      readdevice(Old),
      openread(input,Filename),
      readdevice(input),
      plot_point(-6),
      !,
   readdevice(Old),
   closefile(input).
plot_points(_):-                % coords & data in different files
   wcoord_file(Filename),
   wdata_file(Datafile),
   !,
   readdevice(Old),
   openread(input,Filename),
   openread(datafile,Datafile),
   wdata_position(Pos),
   readdevice(input),
   plot_point(Pos),
   !,
   readdevice(Old),
   closefile(input),
   closefile(datafile).

plot_point(-2):-                % use feature codes from point file
   repeatread(input),
   readterm(plotdata,node(_,Feature,X,Y,_,_)),
   wsetup(Feature,Col,Form,_,Size,_,_,_,_),
   setfillstyle(solid_fill,Col),
   setlinestyle(solid_line,0,norm_width),
   setcolor(Col),
   draw_point(X,Y,Size,Form),
   fail,!.
plot_point(-6):-                % use values from point file
   repeatread(input),
   readterm(plotdata,node(_,_,X,Y,_,Val)),
   value_to_class(Val,Feature),
   wsetup(Feature,Col,Form,_,Size,_,_,_,_),
   setfillstyle(solid_fill,Col),
   setlinestyle(solid_line,0,thick_width),
   setcolor(Col),
   draw_point(X,Y,Size,Form),
   fail,!.
plot_point(2):-                 % use feature codes data file
                            % with coords from polygon file
   repeatread(input),
   readterm(plotdata,node(_,Feature,X,Y,_,_)),
   wsetup(Feature,Col,Form,_,Size,_,_,_,_),
   setfillstyle(solid_fill,Col),
   setlinestyle(solid_line,0,norm_width),
   setcolor(Col),
   draw_point(X,Y,Size,Form),
   fail,!.
plot_point(Num):-               % use values from data file
                            % with coords from polygon file
   Num >= 3,
   repeatread(input),
   readterm(plotdata,polygon(_,_,_,_,X,Y,_)),
   readval(Num,Val),
   value_to_class(Val,Feature),
   wsetup(Feature,Col,Form,_,Size,_,_,_,_),
   setfillstyle(solid_fill,Col),
   setlinestyle(solid_line,0,norm_width),
   setcolor(Col),
   draw_point(X,Y,Size,Form),
   fail,!.
plot_point(_):-                 % on end of coord file
!.
```

```
repeatread(_).
repeatread(File):-not(eof(File)),!,repeatread(File).

readval(1,Val):-
   readdevice(Old),
   readdevice(datafile),
   readterm(plotdata,data(Val,_,_,_,_)),
   readdevice(Old).
readval(3,Val):-
   readdevice(Old),
   readdevice(datafile),
   readterm(plotdata,data(_,_,Val,_,_)),
   readdevice(Old).
readval(4,Val):-
   readdevice(Old),
   readdevice(datafile),
   readterm(plotdata,data(_,_,_,Val,_)),
   readdevice(Old).
readval(5,Val):-
   readdevice(Old),
   readdevice(datafile),
   readterm(plotdata,data(_,_,_,_,Val)),
   readdevice(Old).

feat_look_up(Fcode,Feature):-     % lookup value exists
   wlook_up(Feature,Fcode),
   !.
feat_look_up(Feature,Feature).      % no lookup - try database Fcode

value_to_class(Val,Class):-
   wcurrent_class(F_class),
   fclass_intervals(F_class,[_|Ints]),
   !,
   Classes = [class1,class2,class3,class4,class5,class6,class7,
                        class8,class9,class10],
   value_to_class1(Val,Ints,Classes,Class).

value_to_class1(_,[],_,class0):-
   gmessage("value out of class ranges"),gpause.
value_to_class1(Val,[I1|_],[C1|_],C1):-
   Val <= I1,!.
value_to_class1(Val,[_|Ints],[_|Classes],Class):-
   value_to_class1(Val,Ints,Classes,Class).
```

```
/*****************************************************************/
/*                                                              */
/*    POLIT4.PRO        5/10/94                                 */
/*                                                              */
/*****************************************************************/
/*    modularised version of polit.pro                          */
/*                                                              */
/*    D.F.  30/9/94                                             */
/*****************************************************************/
/*                                                              */
/*    FUNCTIONS                                                 */
/*                                                              */
/*    global predicate sort_adjacent_zones(F_class)             */
/*    reads in .arc file with left & right labels               */
/*    asserts these temporarily & assigns each zone             */
/*    a class different from each neighbour                     */
/*                                                              */
/*    currently 'hard wired' filename                           */
/*                                                              */

project "mapdes"
   include "Globdoms.pro"
   include "tools\\tpreds.pro"
   include "tools\\status.pro"
   include "tools\\menu.pro"
   include "tools\\lineinp.pro"
   include "utils.pro"

CONSTANTS
   gstatus_win          = 63        % window for graphics status line
   gmess_win            = 64        % window for graphics messages

DATABASE - LOCAL
/* in global work
   wcolour(             % relates zone name to group
            SYMBOL,                 % zone
            SYMBOL)                 % group (colour)
*/
   wadjacent(                 % database term for adjacent zones
            SYMBOL,
            SYMBOL)
   wclasses(
            INTEGER,          % number of classes
            SYMBOLLIST)       % class names

PREDICATES
%   sort_adjacent_classes(    % declared in globdoms
%           SYMBOL)                 % feature class
   get_adjacent(        % gets adjacent zones from database
                        % and asserts locally
            SYMBOL)                 % feature class
   repeatread (FILE)
   get_zones(                 % compile list of zones
            SYMBOLLIST)
   assign_initial_colours(    %
            SYMBOLLIST,
            SYMBOLLIST)
   writecolours
   changecolour(SYMBOL,
            SYMBOL)
   sort_adjacent_classes2(
           SYMBOLLIST)
  nextcolour(
           SYMBOL)
  no_clash(
           SYMBOL,
           SYMBOL)

CLAUSES
  sort_adjacent_classes(F_Class):-
     retractall(wcolour(_,_)),
     gstatus("getting zones - please wait"),
     get_adjacent(F_class),
     get_zones(Zones),
     findall(Class,fsymbolism(F_class,Class,_),Classes),
     length_of(Classes,Num),
     assert(wclasses(Num,Classes)),
     !,
     gstatus("assigning symbols to zones - please wait"),
     sort_adjacent_classes2(Zones),
     !,
     writecolours,
     retractall(_,local).

  sort_adjacent_classes2([]):-
     !.
  sort_adjacent_classes2([Z1|Zones]):-
     retractall(wcolour(start,_)),
     wclasses(_,[Start|_]),
     asserta(wcolour(start,Start)),
     nextcolour(Col1),
     retractall(wcolour(Z1,_)),
     no_clash(Z1,Col1),
     assertz(wcolour(Z1,Col1)),
     sort_adjacent_classes2(Zones).

  nextcolour(Colour):-
     wclasses(Num,[Colour|Colours]),
     append(Colours,[Colour],New),
     retractall(wclasses(_,_)),
     asserta(wclasses(Num,New)).
  nextcolour(Colour):-
     wclasses(_,[Start|_]),
     not(wcolour(start,Start)),
     nextcolour(Colour).

  no_clash(Zone,Colour):-
     wadjacent(Zone,Z),
     wcolour(Z,Colour),
     !,
     fail.
  no_clash(Zone,Colour):-
     wadjacent(Z,Zone),
     wcolour(Z,Colour),
     !,
     fail.
  no_clash(_,_).             % succeed - proceed to next zone


  changecolour(Zone,Class):-       % increase colour
     wclasses(Max,Classes),
      listposition(Classes,Class,0,Pos),
      Pos < Max,              % not last on list
      P1 = Pos + 1,
      !,
      list_from_index(Classes,[P1],[C1|_]),
      retractall(wcolour(Zone,_)),
      assertz(wcolour(Zone,C1)).
   changecolour(Zone,_):-           % set colour back to 1
      wclasses(_,[Class|_]),
      retractall(wcolour(Zone,_)),
      assertz(wcolour(Zone,Class)).

   get_adjacent(_):-
      Filename = "c:\\prolog33\\map\\nigeria\\adminbnd.arc",
      file_exist(Filename),
      openread(input,Filename),
      readdevice(input),
      repeatread(input),
      readterm(plotdata,chain(_,"State Boundaries",Right,Left,_,_,_,_)),
      assertz(wadjacent(Right,Left)),
      fail.
   get_adjacent(_):-
      readdevice(keyboard),
%write("got data"),nl,readkey(_),
      closefile(input).

   repeatread(_).
   repeatread(File):-not(eof(File)),!,repeatread(File).

   get_zones(Zones):-
       findall(A,wadjacent(A,_),As),
       findall(B,wadjacent(_,B),Bs),
       append(As,Bs,Cs),
       uniquelist(Cs,Zones).

   assign_initial_colours([],_):-!.
   assign_initial_colours([First|Rest],[G1|Others]):-
      assertz(wcolour(First,G1)),
      append([G1],Others,Update),
      assign_initial_colours(Rest,Update).

   writecolours:-
      openwrite(output,"diags.lis"),
      writedevice(output),
      wcolour(State,Colour),
      nl,write(State,"  ",Colour),
      fail,!.
   writecolours:-
      writedevice(screen),
      closefile(output).
```

```
/*****************************************************************/
/*          7  MODIFY                                            */
/*****************************************************************/

project "mapdes"
include "globdoms.pro"

   include "tools\\tpreds.pro"
   include "tools\\status.pro"
   include "tools\\menu.pro"
   include "tools\\lineinp.pro"
   include "utils.pro"

PREDICATES
%   modify  now in globdoms

CLAUSES
   modify:-
      wquit("y"),!.
   modify:-
      status(1),
      clearmessage,
      makewindow(40,30,7, " MODIFY ",0,0,20,80),
      pause,
      removewindow.
```

```
/***************************************************************/
/*          UTILITIES                                          */
/*    included by call in GLOBDOMS.PRO                         */
/*                                                             */
/*  by D.F. unless other source given                          */
/*                                                             */
/*    06/08/93    many non globally required predicates        */
/*                moved to local modules                       */
/***************************************************************/

/*** Working with LISTS ***/
PREDICATES
   member(integer,integerlist)      /*  checks if variable is a member */
   member(real,reallist)            /*    of a list   */
   member(symbol,symbollist)        /*  Turbo manual p.48 */
   member(string,stringlist)
   member(symbolspec,symbolspeclist)

   append(integerlist,integerlist,integerlist) /*append one list to another*/
   append(symbollist,symbollist,symbollist)     /* Turbo manual p.49 */
   append(stringlist,stringlist,stringlist)
   append(reallist,reallist,reallist)
   append(symbolspeclist,symbolspeclist,symbolspeclist)

%          list   result
   length_of(INTEGERLIST,INTEGER)   % get length of list
   length_of(STRINGLIST,INTEGER)    % calls 3 arity version with 0 to start
   length_of(SYMBOLLIST,INTEGER)
   length_of(REALLIST,INTEGER)
%          list   working  result
   length_of(INTEGERLIST,INTEGER,INTEGER) % get length of list
   length_of(STRINGLIST,INTEGER,INTEGER)  % 3.20 UG p.180
   length_of(SYMBOLLIST,INTEGER,INTEGER)
   length_of(REALLIST,INTEGER,INTEGER)

   remove(integer,integerlist,integerlist) /*remove item from a list */
   remove(symbol,symbollist,symbollist)
   remove(string,stringlist,stringlist)
   remove(real,reallist,reallist)

%  uniquelist(Inlist,   Outlist)
   uniquelist(stringlist,stringlist)      % checks to see that a value
   uniquelist(symbollist,symbollist)      % only appears one in a list

/* replace(Old,New,Oldlist,Newlist)       replaces a member of a list */
   replace(string,string,stringlist,stringlist)
   replace(SYMBOL,SYMBOL,SYMBOLLIST,SYMBOLLIST)
   replace1(string,string,stringlist,stringlist,stringlist)
   replace1(SYMBOL,SYMBOL,SYMBOLLIST,SYMBOLLIST,SYMBOLLIST)

   member_from_index(stringlist,integer,string)
   member_from_index(SYMBOLLIST,INTEGER,SYMBOL)
   member_from_index(REALLIST,INTEGER,REAL)
   member_from_index(INTEGERLIST,INTEGER,INTEGER)

/* list_from_index(Inlist,Numlist,Outlist)  makes outlist from numbered  */
   list_from_index(stringlist,integerlist,stringlist)
   list_from_index(SYMBOLLIST,INTEGERLIST,SYMBOLLIST)
   list_from_index1(stringlist,integerlist,stringlist,stringlist)
   list_from_index1(SYMBOLLIST,INTEGERLIST,SYMBOLLIST,SYMBOLLIST)
   maxinlist(INTEGERLIST,INTEGER)
   maxinlist(REALLIST,REAL)
   maxinlist(INTEGERLIST,INTEGER,INTEGER)
   maxinlist(REALLIST,REAL,REAL)

   mininlist(INTEGERLIST,INTEGER)
   mininlist(REALLIST,REAL)
   mininlist(INTEGERLIST,INTEGER,INTEGER)
   mininlist(REALLIST,REAL,REAL)

   listposition(       % returns position of value on list
               SYMBOLLIST,      % input list of values
               SYMBOL,               % value to be found
               INTEGER,         % working count
               INTEGER)         % position (0=not on list)
   listposition(STRINGLIST,STRING,INTEGER,INTEGER)
   listposition(INTEGERLIST,INTEGER,INTEGER,INTEGER)
   listposition(REALLIST,REAL,INTEGER,INTEGER)

/*** MISC ***/
/* greater_of( X, Y, Ans)         returns Ans as greater of X or Y */
   greater_of(real,real,real)

/* nearest_larger (X,list,Ans)          Ans = first no in ordered list
                                  larger than X          */
   nearest_larger(real,reallist,real)

/* lesser_of(in1,in2,ans)         ans = smaller number           */
   lesser_of(real,real,real)

/***  READING AND WRITING                                   ***/
   write_a_list(integerlist)
   write_a_list(symbollist)
   write_a_list(reallist)
   write_a_list(stringlist)

CLAUSES
   member(X,[X|_]):- !.
   member(X,[_|Tail]) :- member(X,Tail).

   append([],List2,List2):-!.
   append([X|L1],List2,[X|L3]) :- append (L1,List2,L3).

   length_of(List,Result):-
      length_of(List,0,Result).

   length_of([],Result,Result):-!.
   length_of([_|T],Counter,Result):-
      Newcounter=Counter+1,
      !,
      length_of(T,Newcounter,Result).

   remove(_,[],[]):-!.
   remove(Var,Inlist,Inlist):-             % var not member of list - error?
      not(member(Var,Inlist)),!.      % not neccessary but may be useful
                                  % could send a message back
   remove(Var,[Var|Inlist],Inlist):-!.    % var at head of list
   remove(Var,[_|T],Outlist):-
      remove(Var,T,Outlist).

   uniquelist([],[]):-!.
   uniquelist([In|Inlist],Outlist):-            % on list - remove
      member(In,Inlist),!,
      uniquelist(Inlist,Outlist).
   uniquelist([In|Inlist],[In|Outlist]):-       % not on list
      uniquelist(Inlist,Outlist).

   replace(Old,New,[Old|T],Newlist):-!,
      append([New],T,Newlist).
   replace(Old,New,[H|T],Newlist):-
      append([],[H],Hlist),!,
      replace1(Old,New,T,Hlist,Newlist).

   replace1(Old,New,[Old|T],Hlist,Newlist):-!,
      append(Hlist,[New],L1),
      append(L1,T,Newlist).
   replace1(Old,New,[H|T],Hlist,Newlist):-
      append(Hlist,[H],Hlist1),!,
      replace1(Old,New,T,Hlist1,Newlist).

   member_from_index([H|_],1,H):-!.
   member_from_index([_|T],Index,Ans):-
      N = Index - 1,!,
      member_from_index(T,N,Ans).

   list_from_index(_,[],[]):-!.
   list_from_index(Inlist,[H|T],Outlist):-
      list_from_index1(Inlist,[H|T],Outlist,_).

   list_from_index1(Inlist,[H|T],Outlist,Worklist):-
      member_from_index(Inlist,H,Ans),
      list_from_index(Inlist,T,Worklist),!,
      append(Worklist,[Ans],Outlist).

   maxinlist([],0):-!.
   maxinlist([H|Inlist],Max):-
     maxinlist(Inlist,H,Max).
   maxinlist([],Inmax,Inmax):-!.
   maxinlist([H|Inlist],Inmax,Outmax):-
      greater_of(H,Inmax,Newmax),!,
      maxinlist(Inlist,Newmax,Outmax).

   mininlist([],0):-!.
   mininlist([H|Inlist],Min):-
     mininlist(Inlist,H,Min).
   mininlist([],Inmin,Inmin):-!.
   mininlist([H|Inlist],Inmin,Outmin):-
      lesser_of(H,Inmin,Newmin),!,
      mininlist(Inlist,Newmin,Outmin).

   listposition([],_,_,0):-
       !,fail.
   listposition([Val|_],Val,Count,Pos):-
       !,
      Pos = Count + 1.
   listposition([_|T],Val,Count,Pos):-
      Newcount = Count + 1,
      !,
      listposition(T,Val,Newcount,Pos).


   greater_of(X,Y,X):- X >= Y, !.
```

```
   greater_of(_,Y,Y).

   nearest_larger (X,[H|_],H) :- X <= H, !.     % list is ordered low - hi
   nearest_larger (X,[_|T],Ans) :- nearest_larger (X,T,Ans).

   lesser_of(X,Y,Y):- X >= Y, !.
   lesser_of(X,_,X).

   write_a_list([]):-!.
   write_a_list([H|T]) :-
      write(H),nl,!,write_a_list(T).

/***  SCREEN OUTPUT ETC                                     ***/
PREDICATES
   message(string,string)           /* writes a message in window 1 */
   clearmessage                     /* clears message window  */
   ermessage(string,string)         /* beeps and writes message */
   quit                             /* ask if want to quit  */
   pause                            /* wait for space bar to be pressed*/
   status(integer)                  /* changes status line to numbered
                                          message  */

CLAUSES
   message(String1,String2):-
      shiftwindow(Old),
      shiftwindow(1),
      clearwindow,
      write(String1),nl,write(String2),
      shiftwindow(Old).

   clearmessage:-
      shiftwindow(Old),
      shiftwindow(1),
      clearwindow,
      shiftwindow(Old).

   ermessage(String1,String2):-!,
      beep,
      shiftwindow(Old),
      shiftwindow(1),
      clearwindow,
      write(String1),nl,write(String2),
      pause,
      shiftwindow(Old).

   quit:-
      changestatus(Old),
      status(1),
      lineinput(3,10,55,15,15,
          " Do you want to quit the current activity? (y/n) ","N",Ch),
      changestatus(Old),
      Ch = "y",
      asserta(wquit("y")).

   pause:-
% stops processing and waits for space bar to be pressed
      status(Old),!,
      repeat,
      status(3),
      readchar(C), C = ' ',!,
      clearmessage,           % clears any messages after pause
      status(Old).

/* STATUS LINES */
/* 0  for use with main menu when quit not available) */
   status(0) :- !,changestatus(
"F1:Help        Use arrow keys to select, F10 to activate").
/* 1  simple instruction on screen */
   status(1) :- !,changestatus(
"F1:Help
ESC:quit"
   ).
/* 2  single item menu */
   status(2) :- !,changestatus(
"F1:Help F3:Why  Use arrow keys to select choice, F10 to activate.
ESC:quit"
   ).
/* 3  Pause */
   status(3) :- !,changestatus(
"                  Press space bar to continue " ).
/* 4  Multiple choices from menu */
   status(4) :- !,changestatus(
"F1:Help F3:Why  Use RETURN key to select choices, F10 when finished.
ESC:quit"
   ).
/* 5  Data Input */
   status(5) :- !,changestatus(
"F1:Help  F3: Why?      Enter values, press F10 when finished
ESC:quit"
).
/* 6  Request, menu available */
   status(6) :- !,changestatus(
"F1:Help  F2: Menu  F3:Why?
ESC:quit"
).
/* 7  General when no menu available */
   status(7) :- !,changestatus(
"F1:Help  F3:Why?
ESC:quit"
).
/* 8  Answer (solution) on screen */
   status(8) :- !,changestatus(
"F1:Help  F4:How?      Press SPACE BAR to continue               ESC:quit"
).
/* 9  Wait */
   status(9) :- !,changestatus(
"                 PLEASE WAIT "
).

/*************************************************************/
/*    MISC. OPERATIONS                                      */
/*************************************************************/
PREDICATES
   file_exist(STRING)
   failed(STRING)
   rad_deg(REAL,REAL)              % converts from rad to degrees

%  storemap     moved to main     /* store current map specs to KMAPS */
%  load_places        moved to lay2

CLAUSES
   file_exist(File):-
   existfile(File),!.
file_exist(File):-
   concat(File, "  was not found", Outstring),
   ermessage("File Error: the file ",Outstring).

failed(Where):-
   concat("Failed due to unresolved  ", Where, Outstring),
   ermessage(Outstring,""),
   fail.

rad_deg(Rad,Deg):-
   bound(Deg),!,
   Pi = 3.141592653,
   Rad = (Pi * Deg) / 180.
rad_deg(Deg,Deg):-
   failed ("rad_deg : Deg not bound"),!.
```

APPENDIX B

Knowledge base listings

Setup.kba

```
flname("kbasefile","c:\\prolog33\\map\\kbase.kba")
flname("locatefile","c:\\prolog33\\map\\nigeria\\africa.kba")
flname("repfile","c:\\prolog33\\map\\kreps.kba")
flname("symbolfile","c:\\prolog33\\map\\ksymrule.kba")
flname("metafile","c:\\prolog33\\map\\kmeta.kba")
```

Main knowledge base - kbasefile = "kbase.kba"

```
krequired("baseinfo","Other Rivers","Large Rivers",5)
krequired("baseinfo","Large Rivers","Major Rivers",5)
krequired("baseinfo","Major Rivers","Lakes",5)
krequired("baseinfo","Lakes","Lake_fill",5)
krequired("baseinfo","Coastline","Seas",3)
krequired("baseinfo","State Boundaries","International Boundaries",5)
krequired("baseinfo","Tertiary Boundaries","State Boundaries",5)
krequired("baseinfo","Minor Towns","Major Towns",3)
krequired("baseinfo","Major Towns","Capitals",3)
krequired("baseinfo","Other Roads","Highways",5)
krequired("baseinfo","Highways","Main Highways",5)
krequired("baseinfo","Minor Relief","Main Relief",5)
kmember_of("Rivers","Major Rivers")
kmember_of("Rivers","Large Rivers")
kmember_of("Rivers","Other Rivers")
kmember_of("Administrative Boundaries","International Boundaries")
kmember_of("Administrative Boundaries","State Boundaries")
kmember_of("Administrative Boundaries","Tertiary Boundaries")
kmember_of("Settlements","Capitals")
kmember_of("Settlements","Major Towns")
kmember_of("Settlements","Minor Towns")
kmember_of("Relief","Main Relief")
kmember_of("Relief","Minor Relief")
kmember_of("Roads","Main Highways")
kmember_of("Roads","Highways")
kmember_of("Roads","Other Roads")
kmember_of("theme_info","Administrative Areas")
kmember_of("theme_info","Urban population")
kmember_of("theme_info","Rural population")
kmember_of("theme_info","Total population")
kmember_of("theme_info","Population density")
kmember_of("theme_info","Population change")
kmember_of("theme_info","Industrial locations")
kmember_of("theme_info","Agriculture")
kmember_of("theme_info","Geology")
kmember_of("theme_info","Soils")
kmember_of("theme_info","Vegetation")
kmember_of("theme_info","Precipitation")
kmember_of("theme_info","Temperature")
klist("base_info_types",["Coastline","Major Rivers","Large Rivers","Other
  Rivers","Lakes","International Boundaries","State Boundaries","Tertiary
  Boundaries","Capitals","Major Towns","Minor Towns","Urban Areas","Main
  Highways","Highways","Other Roads","Railways","Main Relief","Minor Relief"])
klist("base_info_list",["Coastline","Rivers","Lakes","Administrative
  Boundaries","Settlements","Roads","Railways","Relief"])
klist("theme_info_types",["Administrative
  areas","Agriculture","Geology","Industrial locations","Population
  change","Population density","Precipitation","Rural
  population","Soils","Temperature","Total population","Urban
  population","Vegetation"])
klist("Administrative areas",["Countries","States"])
klist("Agriculture",["Desert","Non-intensive Livestock Farming","Subsistence
  Farming","Commercial Farming","Irrigated Areas","Forest","Unproductive
  Land"])
klist("Geology",["Recent sediments","Sedimentary rocks","Extrusive igneous
  rocks","Intrusive igneous rocks","Metamorphic rocks"])
klist("Industrial locations",["thermal power","hydro
  power","textiles","cement","chemicals","paper","textiles"])
```

```
klist("Population",["Total Population","Population Density","Population
  Change"])
klist("Soils",["Leached red tropical soils","Iron rich savana soils","Rich
  brown tropical soils","Tropical black earths","Brown sub-arid
  soils","Immature soils","Recent alluvium","Raw mineral
  soils","Andosols","Sailine soils"])
klist("Vegetation",["Aquatic grassland swamp","Tropical wooded steppe","Grass
  savanna","Mixed woodland","Tropical savanna woodland","Forest-savanna
  mosaic","Tropical rain forest","Swamp forest","Mangrove swamp"])
kmap_content("basic","map
  type",[10,0,0,0,8,10,0,0,10,0,0,0,0,0,0,0,0,0],[],"baseinfo")
kmap_content("cultural","map
  type",[10,6,3,0,8,10,8,6,10,9,8,6,9,7,5,6,2,0],["Administrative
  areas","Agriculture","Industrial locations","Population change","Population
  density","Rural population","Total population","Urban
  population"],"baseinfo")
kmap_content("physical","map
  type",[10,10,9,7,10,10,6,1,5,3,1,2,4,2,0,2,8,5],["Geology","Precipitation","
  Soils","Temperature","Vegetation"],"baseinfo")
kmap_content("outline","basic",[10,6,3,0,8,10,6,1,5,3,2,2,4,2,0,2,2,0],[],"bas
  einfo")
kmap_content("topographic","basic",[10,10,8,6,10,10,8,6,10,8,6,6,10,8,4,8,10,6
  ],[],"baseinfo")
kmap_content("political","cultural",[10,6,3,0,9,10,9,6,10,9,8,6,8,7,5,6,2,0],[
  "Administrative areas"],"baseinfo")
kmap_content("population","cultural",[10,6,3,0,8,10,9,7,10,9,8,6,9,7,5,6,2,0],
  ["Population change","Population density","Rural population","Total
  population","Urban population"],"baseinfo")
kmap_content("economic","cultural",[10,6,3,0,8,10,8,6,10,9,8,6,9,7,5,6,2,0],["
  Agriculture","Industrial locations"],"baseinfo")
kmap_content("settlements","cultural",[10,6,3,0,8,10,9,6,10,10,9,8,10,7,5,8,4,
  0],[],"baseinfo")
kmap_content("urban","population",[10,6,3,0,8,10,9,6,5,4,2,3,9,7,5,6,2,0],["Ur
  ban population"],"baseinfo")
kmap_content("rural","population",[10,6,3,0,8,10,9,8,10,9,8,6,9,7,5,6,2,0],["R
  ural population"],"baseinfo")
kmap_content("industries","economic",[10,6,3,0,8,10,8,6,10,9,8,6,9,7,5,6,2,0],
  ["Industrial locations"],"baseinfo")
kmap_content("agriculture","economic",[10,10,9,7,10,10,6,1,5,3,1,2,4,2,0,2,8,5
  ],["Agriculture"],"baseinfo")
kmap_content("communications","economic",[10,6,3,0,10,10,8,6,10,9,8,6,9,7,5,6,
  2,0],[],"baseinfo")
kmap_content("relief","physical",[10,10,9,7,10,10,6,1,5,3,1,2,4,2,0,2,10,8],[]
  ,"baseinfo")
kmap_content("land_cover","physical",[10,10,9,7,10,10,6,1,5,3,1,2,4,2,0,2,8,5]
  ,["Agriculture","Geology","Soils","Vegetation"],"baseinfo")
kmap_content("climate","physical",[10,10,9,7,10,10,6,1,5,3,1,2,4,2,0,2,8,5],["
  Precipitation","Temperature"],"baseinfo")
kmap_content("soils","land_cover",[10,10,9,7,10,10,6,1,5,3,1,2,4,2,0,2,8,5],["
  Soils"],"baseinfo")
kmap_content("vegetation","land_cover",[10,10,9,7,10,10,6,1,5,3,1,2,4,2,0,2,8,
  5],["Vegetation"],"baseinfo")
kmap_content("geology","land_cover",[10,10,9,7,10,10,6,1,5,3,1,2,4,2,0,2,8,5],
  ["Geology"],"baseinfo")
kmap_content("precipitation","climate",[10,10,9,7,10,10,6,1,5,3,1,2,4,2,0,2,8,
  5],["Precipitation"],"baseinfo")
kmap_content("temperature","climate",[10,10,9,7,10,10,6,1,5,3,1,2,4,2,0,2,8,5]
  ,["Temperature"],"baseinfo")
klevel_of_detail("author","overview","screen",4)
klevel_of_detail("author","analysis","screen",8)
klevel_of_detail("general","overview","screen",2)
klevel_of_detail("general","analysis","screen",6)
klevel_of_detail("specialist","overview","screen",4)
klevel_of_detail("specialist","analysis","screen",8)
kscale_range(2000000,15000000,"")
```

Cartographic representation knowledge base - repfile = "kreps.kba"

```
ksubclass("Rivers","Major Rivers")
ksubclass("Rivers","Large Rivers")
ksubclass("Rivers","Other Rivers")
ksubclass("Administrative Boundaries","International Boundaries")
ksubclass("Administrative Boundaries","State Boundaries")
ksubclass("Administrative Boundaries","Tertiary Boundaries")
ksubclass("Settlements","Capitals")
ksubclass("Settlements","Major Towns")
ksubclass("Settlements","Minor Towns")
ksubclass("Relief","Main Relief")
ksubclass("Relief","Minor Relief")
ksubclass("Roads","Main Highways")
ksubclass("Roads","Highways")
ksubclass("Roads","Other Roads")
krepresentation_type(11,"dot distribution - individuals")
krepresentation_type(12,"dot distribution - groups")
krepresentation_type(21,"categorised points")
krepresentation_type(31,"ranked points")
krepresentation_type(41,"proportional - classed")
krepresentation_type(42,"proportional - graduated")
krepresentation_type(43,"proportional - bi-polar")
krepresentation_type(51,"multivariable - fixed size")
krepresentation_type(52,"multivariable - classed")
krepresentation_type(53,"multivariable - graduated")
krepresentation_type(61,"spot values")
krepresentation_type(62,"irregular value network")
krepresentation_type(63,"grid surface values")
krepresentation_type(111,"boundaries - one level")
krepresentation_type(112,"boundaries - hierarchy")
krepresentation_type(121,"network - link & node")
krepresentation_type(122,"network - branching")
krepresentation_type(131,"isolines")
krepresentation_type(141,"flowlines")
krepresentation_type(151,"misc. tangible lines")
krepresentation_type(211,"isolated areas")
krepresentation_type(221,"unclassed areas - one level")
krepresentation_type(222,"unclassed areas - hierarchy")
krepresentation_type(231,"categorical - one level")
krepresentation_type(232,"categorical - hierarchy")
krepresentation_type(241,"graded series - unipolar")
krepresentation_type(242,"graded series - bipolar")
krepresentation_type(243,"graded series - bivariate")
krepresentation_type(251,"layers - uniploar")
krepresentation_type(252,"layers - bipolar")
krepresentation_type(261,"hypsometric layers")
krep("rep_type","Coastline","boundaries - one level",5)
krep("rep_type","Seas","isolated areas",5)
krep("rep_type","Rivers","network - branching",5)
krep("rep_type","Lakes","boundaries - one level",5)
krep("rep_type","Lake_fill","isolated areas",5)
krep("rep_type","Administrative Boundaries","boundaries - hierarchy",5)
krep("rep_type","Settlements","ranked points",5)
krep("rep_type","Urban Areas","isolated areas",5)
krep("rep_type","Roads","network - link & node",5)
krep("rep_type","Railways","network - link & node",5)
krep("rep_type","Relief","hypsometric layers",3)
krep("rep_type","Relief","isolines",2)
krep("rep_type","Relief","isolated areas",1)
krep("rep_type","Countries","unclassed areas - one level",5)
krep("rep_type","Administrative areas","unclassed areas - hierarchy",5)
krep("rep_type","Agriculture","categorical - one level",5)
krep("rep_type","Geology","categorical - one level",5)
krep("rep_type","Soils","categorical - one level",5)
krep("rep_type","Vegetation","categorical - one level",5)
krep("rep_type","Total population","proportional - classed",4)
krep("rep_type","Total population","proportional - graduated",3)
krep("rep_type","Population density","graded series - unipolar",4)
krep("rep_type","Population density","graded series - bipolar",2)
krep("rep_type","Population change","graded series - bipolar",5)
krep("rep_type","Urban population","proportional - classed",4)
krep("rep_type","Urban population","proportional - graduated",3)
krep("rep_type","Rural population","graded series - unipolar",4)
krep("rep_type","Rural population","graded series - bipolar",2)
krep("rep_type","Industrial locations","categorised points",5)
krep("rep_type","Precipitation","layers - unipolar",4)
krep("rep_type","Precipitation","layers - bipolar",3)
krep("rep_type","Precipitation","isolines",1)
krep("rep_type","Temperature","layers - bipolar",4)
krep("rep_type","Temperature","layers - unipolar",3)
krep("rep_type","Temperature","isolines",1)
kconflict("rep_type","dot distribution - individuals","dot distribution -
  groups")
kconflict("rep_type","dot distribution - individuals","spot values")
kconflict("rep_type","dot distribution - individuals","irregular value
  network")
kconflict("rep_type","dot distribution - individuals","grid surface values")
kconflict("rep_type","dot distribution - groups","spot values")
kconflict("rep_type","dot distribution - groups","irregular value network")
kconflict("rep_type","dot distribution - groups","grid surface values")
kconflict("rep_type","proportional - classed","proportional - classed")
kconflict("rep_type","proportional - classed","proportional - graduated")
kconflict("rep_type","proportional - classed","proportional - bipolar")
kconflict("rep_type","proportional - classed","multivariate - fixed size")
kconflict("rep_type","proportional - classed","multivariate - classed")
kconflict("rep_type","proportional - classed","multivariate - graduated")
kconflict("rep_type","proportional - graduated","proportional - graduated")
kconflict("rep_type","proportional - graduated","proportional - bipolar")
kconflict("rep_type","proportional - graduated","multivariate - fixed size")
kconflict("rep_type","proportional - graduated","multivariate - classed")
kconflict("rep_type","proportional - graduated","multivariate - graduated")
kconflict("rep_type","proportional - bipolar","proportional - bipolar")
kconflict("rep_type","multivariate - fixed size","multivariate - fixed size")
kconflict("rep_type","multivariate - fixed size","multivariate - classed")
kconflict("rep_type","multivariate - fixed size","multivariate - graduated")
kconflict("rep_type","multivariate - classed","multivariate - classed")
kconflict("rep_type","multivariate - classed","multivariate - graduated")
kconflict("rep_type","multivariate - graduated","multivariate - graduated")
kconflict("rep_type","spot values","irregular value network")
kconflict("rep_type","spot values","grid surface values")
kconflict("rep_type","irregular value network","irregular value network")
kconflict("rep_type","irregular value network","grid surface values")
kconflict("rep_type","grid surface values","grid surface values")
kconflict("rep_type","unclassed areas - one level","unclassed areas - one
  level")
kconflict("rep_type","unclassed areas - one level","unclassed areas -
  hierarchy")
kconflict("rep_type","unclassed areas - one level","categorical - one level")
kconflict("rep_type","unclassed areas - one level","categorical - hierarchy")
kconflict("rep_type","unclassed areas - one level","graded series - unipolar")
kconflict("rep_type","unclassed areas - one level","graded series - bipolar")
kconflict("rep_type","unclassed areas - one level","graded series -
  bivariate")
kconflict("rep_type","unclassed areas - one level","layers - unipolar")
kconflict("rep_type","unclassed areas - one level","layers - bipolar")
kconflict("rep_type","unclassed areas - one level","hypsometric layers")
kconflict("rep_type","unclassed areas - hierarchy","unclassed areas -
  hierarchy")
kconflict("rep_type","unclassed areas - hierarchy","categorical - one level")
kconflict("rep_type","unclassed areas - hierarchy","categorical - hierarchy")
kconflict("rep_type","unclassed areas - hierarchy","graded series - unipolar")
kconflict("rep_type","unclassed areas - hierarchy","graded series - bipolar")
kconflict("rep_type","unclassed areas - hierarchy","graded series -
  bivariate")
kconflict("rep_type","unclassed areas - hierarchy","layers - unipolar")
kconflict("rep_type","unclassed areas - hierarchy","layers - bipolar")
kconflict("rep_type","unclassed areas - hierarchy","hypsometric layers")
kconflict("rep_type","categorical - one level","categorical - one level")
kconflict("rep_type","categorical - one level","categorical - hierarchy")
kconflict("rep_type","categorical - one level","graded series - unipolar")
kconflict("rep_type","categorical - one level","graded series - bipolar")
kconflict("rep_type","categorical - one level","graded series - bivariate")
kconflict("rep_type","categorical - one level","layers - unipolar")
kconflict("rep_type","categorical - one level","layers - bipolar")
kconflict("rep_type","categorical - one level","hypsometric layers")
kconflict("rep_type","categorical - hierarchy","categorical - hierarchy")
kconflict("rep_type","categorical - hierarchy","graded series - unipolar")
kconflict("rep_type","categorical - hierarchy","graded series - bipolar")
kconflict("rep_type","categorical - hierarchy","graded series - bivariate")
kconflict("rep_type","categorical - hierarchy","layers - unipolar")
kconflict("rep_type","categorical - hierarchy","layers - bipolar")
kconflict("rep_type","categorical - hierarchy","hypsometric layers")
kconflict("rep_type","graded series - unipolar","graded series - unipolar")
kconflict("rep_type","graded series - unipolar","graded series - bipolar")
kconflict("rep_type","graded series - unipolar","graded series - bivariate")
kconflict("rep_type","graded series - unipolar","layers - unipolar")
kconflict("rep_type","graded series - unipolar","layers - bipolar")
kconflict("rep_type","graded series - unipolar","hypsometric layers")
kconflict("rep_type","graded series - bipolar","graded series - bipolar")
kconflict("rep_type","graded series - bipolar","graded series - bivariate")
kconflict("rep_type","graded series - bipolar","layers - unipolar")
kconflict("rep_type","graded series - bipolar","layers - bipolar")
kconflict("rep_type","graded series - bipolar","hypsometric layers")
kconflict("rep_type","graded series - bivariate","graded series - bivariate")
kconflict("rep_type","graded series - bivariate","layers - unipolar")
kconflict("rep_type","graded series - bivariate","layers - bipolar")
kconflict("rep_type","graded series - bivariate","hypsometric layers")
kconflict("rep_type","layers - unipolar","layers - unipolar")
kconflict("rep_type","layers - unipolar","layers - bipolar")
kconflict("rep_type","layers - unipolar","hypsometric layers")
kconflict("rep_type","layers - bipolar","layers - bipolar")
kconflict("rep_type","layers - bipolar","hypsometric layers")
kconflict("rep_type","hypsometric layers","hypsometric layers")
```

Symbolisation knowledge base - symbolfile = "ksymrule.kba"

```
ksymbolspec("Coastline","Coastline",symbolspec("line","blue","dark","low","con
   tinuous","",0,"fine",0))
ksymbolspec("Coastline","Coastline",symbolspec("line","black","dark","low","co
   ntinuous","",0,"fine",0))
ksymbolspec("Seas","Seas",symbolspec("area","cyan","light","low","solid","",0,
   "",0))
ksymbolspec("Seas","Seas",symbolspec("area","blue","light","low","solid","",0,
   "",0))
ksymbolspec("Seas","Seas",symbolspec("area","grey","light","low","solid","",0,
   "",0))
ksymbolspec("Administrative Boundaries","International
   Boundaries",symbolspec("line","grey","dark","low","chain","chain",0,"thick",
   0))
ksymbolspec("Administrative Boundaries","International
   Boundaries",symbolspec("line","grey","light","low","chain","chain",0,"thick"
   ,0))
ksymbolspec("Administrative Boundaries","International
   Boundaries",symbolspec("line","red","mid","mid","chain","chain",0,"thick",0)
   )
ksymbolspec("Administrative Boundaries","State
   Boundaries",symbolspec("line","black","dark","low","chain","chain",0,"fine",
   0))
ksymbolspec("Administrative Boundaries","State
   Boundaries",symbolspec("line","grey","dark","low","chain","chain",0,"fine",0
   ))
ksymbolspec("Administrative Boundaries","State
   Boundaries",symbolspec("line","grey","light","low","chain","chain",0,"fine",
   0))
ksymbolspec("Administrative Boundaries","Tertiary
   Boundaries",symbolspec("line","grey","dark","low","dotted","dotted",0,"fine"
   ,0))
ksymbolspec("Administrative Boundaries","Tertiary
   Boundaries",symbolspec("line","grey","light","low","dotted","dotted",0,"fine
   ",0))
ksymbolspec("Rivers","Major
   Rivers",symbolspec("line","blue","dark","mid","continuous","",0,"fine",0))
ksymbolspec("Rivers","Major
   Rivers",symbolspec("line","black","dark","low","continuous","",0,"fine",0))
ksymbolspec("Rivers","Large
   Rivers",symbolspec("line","blue","dark","mid","continuous","",0,"fine",0))
ksymbolspec("Rivers","Large
   Rivers",symbolspec("line","black","dark","low","continuous","",0,"fine",0))
ksymbolspec("Rivers","Other
   Rivers",symbolspec("line","blue","dark","mid","continuous","",0,"fine",0))
ksymbolspec("Rivers","Other
   Rivers",symbolspec("line","black","dark","low","continuous","",0,"fine",0))
ksymbolspec("Railways","Railways",symbolspec("line","grey","dark","low","conti
   nuous","",0,"medium",0))
ksymbolspec("Railways","Railways",symbolspec("line","black","dark","low","cont
   inuous","",0,"medium",0))
ksymbolspec("Relief","contours",symbolspec("line","brown","mid","low","continu
   ous","",0,"fine",0))
kassociate_with("Seas","Lake_fill","symbol",8)
kassociate_with("Coastline","Rivers","hue",8)
kassociate_with("Coastline","Seas","hue",8)
kassociate_with("Coastline","Lakes","symbol",7)
kassociate_with("Lakes","Rivers","hue",10)
kassociate_with("Lakes","Rivers","symbol",8)
kassociate_with("Lake_fill","Lakes","hue",9)
```

```
kconvention("Seas","","hue","blue",9)
kconvention("Seas","","hue","cyan",8)
kconvention("Seas","","hue","white",7)
kconvention("Seas","","hue","grey",6)
kconvention("Seas","","lightness","light",8)
kconvention("Coastline","","hue","blue",9)
kconvention("Lake_fill","","hue","blue",9)
kconvention("Lake_fill","","hue","cyan",8)
kconvention("Lake_fill","","hue","white",7)
kconvention("Lake_fill","","hue","grey",6)
kconvention("Lake_fill","","lightness","light",8)
kconvention("Lake_fill","","lightness","mid",6)
kconvention("Rivers","","hue","blue",9)
kconvention("Rivers","","lightness","dark",7)
kconvention("Rivers","","form","continuous",10)
kconvention("Lakes","","hue","blue",9)
kconvention("Lakes","","lightness","dark",7)
kconvention("Lakes","","form","continuous",10)
kconvention("Administrative Boundaries","","form","chain",8)
kconvention("Administrative Boundaries","","form","dash",6)
kconvention("Administrative Boundaries","","hue","red",8)
kconvention("Administrative Boundaries","","hue","black",7)
kconvention("Administrative Boundaries","","hue","grey",6)
kconvention("Administrative Boundaries","International
  Boundaries","dimension","thick",8)
kconvention("Administrative Boundaries","State
  Boundaries","dimension","fine",9)
kconvention("Urban areas","","hue","red",8)
kconvention("Urban areas","","hue","grey",7)
kconvention("Urban areas","","hue","yellow",6)
kconvention("Settlements","Capitals","form","square",9)
kconvention("Settlements","Major Towns","form","dot",9)
kconvention("settlements","Major Towns","form","box",6)
kconvention("Settlements","Minor Towns","form","circle",9)
kconvention("Settlements","Minor Towns","form","dot",7)
kconvention("Settlements","Capitals","dimension","medium",9)
kconvention("Settlements","Major Towns","dimension","small",9)
kconvention("Settlements","Minor Towns","dimension","v_small",9)
kconvention("Settlements","","hue","magenta",9)
kconvention("Settlements","","hue","red",8)
kconvention("Settlements","","hue","black",7)
kconvention("Settlements","","hue","yellow",6)
kconvention("Roads","","form","continuous",10)
kconvention("Roads","","hue","red",8)
kconvention("Roads","","hue","yellow",7)
kconvention("Roads","","hue","grey",6)
kconvention("Railways","","hue","black",10)
kconvention("Railways","","hue","grey",8)
kconvention("Railways","","form","continuous",10)
kconvention("Relief","Contours","hue","brown",8)
kconvention("Population","","hue","magenta",9)
kconvention("Population","","hue","red",8)
kconvention("Population","Total","form","circle",8)
kconvention("Population","Density","form","tint",10)
kconvention("graded series - unipolar","","hue","green",8)
kconvention("graded series - unipolar","","hue","magenta",7)
kconvention("graded series - unipolar","","hue","purple",7)
kconvention("graded series - unipolar","","hue","red",6)
kconvention("graded series - unipolar","","hue","blue",5)
kconvention("graded series - unipolar","","hue","black",4)
```

```
ksymbolset("equiv_area_colours","mid
   tones","",[symbolspec("area","green","mid","low","solid","",0,"",0),symbolsp
   ec("area","cyan","mid","low","solid","",0,"",0),symbolspec("area","brown","m
   id","low","solid","",0,"",0),symbolspec("area","orange","light","low","solid
   ","",0,"",0),symbolspec("area","magenta","mid","low","solid","",0,"",0),symb
   olspec("area","red","light","low","solid","",0,"",0),symbolspec("area","blue
   ","mid","mid","solid","",0,"",0),symbolspec("area","grey","light","low","sol
   id","",0,"",0)])
ksymbolset("equiv_area_colours","light
   tones","",[symbolspec("area","green","light","mid","solid","",0,"",0),symbol
   spec("area","yellow","pale","mid","solid","",0,"",0),symbolspec("area","oran
   ge","light","mid","solid","",0,"",0),symbolspec("area","red","light","mid","
   solid","",0,"",0),symbolspec("area","magenta","pale","mid","solid","",0,"",0
   ),symbolspec("area","cyan","pale","mid","solid","",0,"",0),symbolspec("area"
   ,"blue","light","mid","solid","",0,"",0),symbolspec("area","grey","light","l
   ow","solid","",0,"",0)])
ksymbolset("equiv_area_colours","light, bright
   colours","",[symbolspec("area","green","light","high","solid","",0,"",0),sym
   bolspec("area","cyan","light","high","solid","",0,"",0),symbolspec("area","y
   ellow","light","high","solid","",0,"",0),symbolspec("area","orange","light",
   "mid","solid","",0,"",0),symbolspec("area","red","mid","high","solid","",0,"
   ",0),symbolspec("area","magenta","mid","high","solid","",0,"",0),symbolspec(
   "area","blue","mid","high","solid","",0,"",0)])
ksymbolset("minor_relief_layers","","land tone colours for
   relief",[symbolspec("area","green","mid","mid","solid","",0,"",0),symbolspec
   ("area","green","light","mid","solid","",0,"",0),symbolspec("area","orange",
   "light","mid","solid","",0,"",0),symbolspec("area","brown","mid","low","soli
   d","",0,"",0),symbolspec("area","brown","mid","mid","solid","",0,"",0)])
ksymbolset("main_relief_layers","","land tone colours for simple
   relief",[symbolspec("area","green","mid","low","solid","",0,"",0),symbolspec
   ("area","brown","mid","low","solid","",0,"",0),symbolspec("area","brown","mi
   d","mid","solid","",0,"",0)])
ksymbollist("point_colours",["red","magenta","green","blue","brown","black"])
ksymbollist("line_colours",["black","red","green","magenta"])
ksymbollist("point_forms",["geometric_points","complex_points","pictorial_poin
   ts","subdivided"])
ksymbollist("line_forms",["continuous","broken_line","cased_line","complex_lin
   e"])
ksymbollist("area_forms",["solid","tint","pattern"])
ksymbollist("geometric_points",["square","dot","box","circle","cross","plus"])
ksymbollist("Settlements",["square","box","dot","circle"])
ksymbollist("ranked_points",["square","box","dot","circle"])
ksymbollist("broken_lines",["dotted","chain","dashed"])
ksymbollist("graduated2",["grad2","grad4"])
ksymbollist("graduated2",["grad1","grad3"])
ksymbollist("graduated2",["grad3","grad5"])
ksymbollist("graduated3",["grad1","grad3","grad5"])
ksymbollist("graduated3",["grad2","grad4","grad6"])
ksymbollist("graduated3",["grad3","grad5","grad7"])
ksymbollist("graduated4",["grad1","grad3","grad5","grad7"])
ksymbollist("graduated4",["grad1","grad2","grad3","grad4"])
ksymbollist("graduated4",["grad2","grad4","grad6","grad8"])
ksymbollist("graduated5",["grad1","grad2","grad3","grad4","grad5"])
ksymbollist("graduated5",["grad0","grad1","grad2","grad3","grad4"])
ksymbollist("graduated5",["grad1","grad3","grad5","grad7","grad9"])
ksymbollist("graduated6",["grad1","grad2","grad3","grad4","grad5","grad6"])
ksymbollist("graduated6",["grad0","grad1","grad2","grad3","grad4","grad5"])
ksymbollist("graduated7",["grad1","grad2","grad3","grad4","grad5","grad6","gra
   d7"])
ksymbollist("graduated8",["grad1","grad2","grad3","grad4","grad5","grad6","gra
   d7","grad8"])
```

```
ksymbollist("graduated9",["grad1","grad2","grad3","grad4","grad5","grad6","gra
   d7","grad8","grad9"])
ksymbollist("graduated10",["grad0","grad1","grad2","grad3","grad4","grad5","gr
   ad6","grad7","grad8","grad9"])
ksymbollist("choro_tints2",["t3","t5"])
ksymbollist("choro_tints3",["t3","t5","t7"])
ksymbollist("choro_tints4",["t3","t5","t7","solid"])
ksymbollist("choro_tints5",["t2","t3","t5","t7","solid"])
ksymbollist("choro_tints6",["t2","t3","t4","t5","t7","solid"])
ksymbollist("choro_tints7",["t2","t3","t4","t5","t6","t8","solid"])
ksymbollist("choro_tints8",["t1","t2","t3","t4","t5","t6","t7","t8","solid"])
kstringlist("class_names2",["class1","class2"])
kstringlist("class_names3",["class1","class2","class3"])
kstringlist("class_names4",["class1","class2","class3","class4"])
kstringlist("class_names5",["class1","class2","class3","class4","class5"])
kstringlist("class_names6",["class1","class2","class3","class4","class5","clas
   s6"])
kstringlist("class_names7",["class1","class2","class3","class4","class5","clas
   s6","class7"])
kstringlist("class_names8",["class1","class2","class3","class4","class5","clas
   s6","class7","class8"])
kstringlist("class_names9",["class1","class2","class3","class4","class5","clas
   s6","class7","class8","class9"])
kstringlist("class_names10",["class1","class2","class3","class4","class5","cla
   ss6","class7","class8","class9","class10"])
kstringlist("group_names2",["group1","group2"])
kstringlist("group_names3",["group1","group2","group3"])
kstringlist("group_names4",["group1","group2","group3","group4"])
kstringlist("group_names5",["group1","group2","group3","group4","group5"])
kstringlist("group_names6",["group1","group2","group3","group4","group5","grou
   p6"])
kstringlist("group_names7",["group1","group2","group3","group4","group5","grou
   p6","group7"])
kstringlist("group_names8",["group1","group2","group3","group4","group5","grou
   p6","group7","group8"])
kstringlist("group_names9",["group1","group2","group3","group4","group5","grou
   p6","group7","group8","group9"])
kstringlist("group_names10",["group1","group2","group3","group4","group5","gro
   up6","group7","group8","group9","group10"])
klevel("","unclassed areas - one level",100)
klevel("Countries","unclassed areas - one level",101)
klevel("","unclassed areas - hierarchy",100)
klevel("Administrative areas","unclassed areas - hierarchy",102)
klevel("","categorical - one level",110)
klevel("Agriculture","categorical - one level",111)
klevel("Geology","categorical - one level",111)
klevel("Soils","categorical - one level",111)
klevel("Vegetation","categorical - one level",111)
klevel("","graded series - unipolar",120)
klevel("","graded series - bipolar",120)
klevel("","graded series - bipolar",120)
klevel("Population density","graded series - unipolar",121)
klevel("Population density","graded series - bipolar",121)
klevel("Population change","graded series - bipolar",121)
klevel("Rural population","graded series - unipolar",121)
klevel("Rural population","graded series - bipolar",121)
klevel("","layers - unipolar",130)
klevel("","layers - bipolar",130)
klevel("Precipitation","layers - unipolar",131)
klevel("Precipitation","layers - bipolar",131)
klevel("Temperature","layers - bipolar",131)
klevel("Temperature","layers - unipolar",131)
klevel("Relief","hypsometric layers",141)
klevel("","isolated areas",150)
klevel("Relief","isolated areas",151)
klevel("Urban Areas","isolated areas",152)
klevel("Seas","isolated areas",158)
klevel("Rivers","network - branching",190)
klevel("Lake_fill","isolated areas",191)
klevel("","boundaries - one level",200)
klevel("Coastline","boundaries - one level",201)
klevel("Lakes","boundaries - one level",202)
klevel("Relief","isolines",204)
klevel("Administrative Boundaries","boundaries - hierarchy",210)
klevel("Administrative Boundaries","boundaries - hierarchy",210)
klevel("","network - link & node",220)
klevel("Railways","network - link & node",221)
klevel("Road casings","network - link & node",222)
klevel("Roads","network - link & node",223)
klevel("","isolines",231)
klevel("Precipitation","isolines",231)
klevel("Temperature","isolines",231)
klevel("","proportional - classed",240)
klevel("","proportional - graduated",240)
klevel("Total population","proportional - classed",241)
klevel("Total population","proportional - graduated",241)
klevel("Urban population","proportional - classed",242)
klevel("Urban population","proportional - graduated",242)
klevel("","ranked points",251)
klevel("Settlements","ranked points",251)
klevel("","categorised points",260)
klevel("Industrial locations","categorised points",261)
```

APPENDIX C

Other listings

1. Lookup.pro - look-up table for converting from symbol specifications to BGI values for plotting

```
DOMAINS
integerlist = integer*
/*
CONSTANTS % declared in BGI.pre & Graph.pre

 % Colors for setpalette and setallpalette
 black          =      0        /* dark colors */
 blue           =      1
 green          =      2
 cyan           =      3
 red            =      4
 magenta        =      5
 brown          =      6
 lightgray      =      7
 darkgray       =      8        /* Light colors */
 lightblue      =      9
 lightgreen     =      10
 lightcyan      =      11
 lightred       =      12
 lightmagenta=         13
 yellow         =      14
 white          =      15

 % Line styles for get/setlinestyle
 solid_line  =         0
 dotted_line =         1
 center_line =         2
 dashed_line  =        3
 userbit_line =        4        /* User defined line style */

 % Line widths for get/setlinestyle
 norm_width  =         1
 thick_width  =        3

 % Fill styles for get/setfillstyle
 empty_fill            = 0      /* Fills area in background color */
 solid_fill            = 1      /* Fills area in solid fill color */
 line_fill             = 2      /* --- fill */
 ltslash_fill          = 3      /* /// fill */
 slash_fill            = 4      /* /// fill with thick lines */
 bkslash_fill          = 5      /* \\\ fill with thick lines */
 ltbkslash_fill        = 6      /* \\\ fill */
 hatch_fill            = 7      /* Light hatch fill */
 xhatch_fill           = 8      /* Heavy cross hatch fill */
 interleave_fill       = 9      /* Interleaving line fill */
 wide_dot_fill         = 10     /* Widely spaced dot fill */
 close_dot_fill        = 11     /* Closely spaced dot fill */
 user_fill             = 12     /* User defined fill */

 % point symbols for toolbox draw_point - size specified in pixels
        plot_pix     = 0      % Single pixel
        plot_circle  = 1       % Empty circle
        plot_dot     = 2      % Filled circle
        plot_box    = 3     % Empty rectangle
        plot_fill   = 4     % Filled rectangle
        plot_plus   = 5     % Plus (+) symbol
        plot_cross  = 6     % Cross (x) symbol
*/

/* SYMBOLISATION LOOK UP TABLES */

/*
  HUE
                magenta
                red
                yellow
                green
                cyan
                blue
                black
                grey
                brown
                orange
                purple
  LIGHTNESS
                pale
                light
                mid
                dark
  SATURATION
                low
                mid
                high
  FORM - AREAS
                solid
                tint
                pattern
        - LINES
                continuous
                dotted
                dashed
                chain
                cased
                complex
        - POINTS                        FORM_CODE
                geometric       -       circle
                                        dot
                                        square
                                        box
                                        cross
                                        plus
                                        % star
                combined
                pictorial
                subdivided
  ORIENTATION
```

```
            - angle of rotation of point symbols or area patterns
  DIMENSION
       - LINES
            fine
            medium
            thick
       - POINTS
            pixel
            v_small
            small
            medium
            large,
            v_large

*/

DATABASE - LOOKUP

  kcolour_table(              % conversion from descriptive to BGI colours
                              % for area fill
            SYMBOL,                % symbolic name for colour
            SYMBOL,                % hue
            SYMBOL,                % lightness
            SYMBOL,                % saturation ?? brightness?
            INTEGER,               % colour number
            INTEGER,               % fill style
            SYMBOL).               % fill pattern - name

  kpattern     (              % look up table for pattern definitions
            SYMBOL,                % pattern name
            INTEGERLIST)           % 8 element list describing pattern

  kline_colour(               % conversion from descriptive to
                              % BGI line colours
            SYMBOL,                % symbolic name for colour
            SYMBOL,                % hue
            SYMBOL,                % lightness
            SYMBOL,                % saturation ?? brightness?
            INTEGER).              % colour number

  kline_form(                 % conversion from descriptive to BGI val
            SYMBOL,                % form code
            SYMBOL,                % form
            INTEGER,               % form value
            INTEGER)               % form style if value = 4 (user)

  kline_width(                % conversion from descriptive to BGI value
            SYMBOL,                % width code
            INTEGER)               % value

  kpoint_form(                % conversion from descriptive to BGI
            SYMBOL,                % form code
            SYMBOL,                % form
```

```
                INTEGER)                % BGI form value

  kpoint_size(                  % conversion from descriptive to BGI value
                SYMBOL,                 % size code
                INTEGER)                % BGI value (rad in pixels)

  klookup(                      % general purpose lookup values
                SYMBOL,                 % feature / name
                SYMBOL,                 % input 1
                SYMBOL,                 % input 2
                SYMBOL,                 % output 1
                SYMBOL)                 % output 2

CLAUSES

% Assumptions made for this colour set:

%       0 background replaced by   EGA 63 - white
%       4 red        replaced by   EGA 36 - true red
%       12 lightred  replaced by   EGA 38 - orange
%       15 white     replaced by   EGA 0  - black

  kcolour_table( black,      black,dark,low,      15,1,"").        % BGI white
  kcolour_table( blue,       blue, dark,  low,    1,12,"gray75").
  kcolour_table( blue,       blue, dark,  mid,    1,1,"").         %blue
% kcolour_table( blue,       blue, dark,  high
  kcolour_table( blue,       blue, mid,   low,    1,12,"gray50").
  kcolour_table( blue,       blue, mid,   mid,    9,12,"gray75").
  kcolour_table( blue,       blue, mid,   high,   9,1,"").         %lightblue
  kcolour_table( blue,       blue, light, low,    1,12,"gray25").
  kcolour_table( blue,       blue, light, mid,    9,12,"gray75").
  kcolour_table( blue,       blue, light, high,   9,1,"").
  kcolour_table( blue,       blue, pale,  low,    9,12,"gray25").
  kcolour_table( blue,       blue, pale,  mid,    9,12,"gray50").
% kcolour_table( blue,       blue, pale,  high,   11,9,"").
% kcolour_table( green,      green, dark,  low,
% kcolour_table( green,      green, dark,  mid
  kcolour_table( green,      green, mid,   low,   2,12,"gray75").
  kcolour_table( green,      green, mid,   mid,   2,1,"").         %green
% kcolour_table( green,      green, mid,   high,
  kcolour_table( green,      green, light, low,   2,12,"gray50").
  kcolour_table( green,      green, light, mid,   10,12,"gray75").
  kcolour_table( green,      green, light, high,  10,1,"").        % lightgreen
  kcolour_table( green,      green, pale,  low,   10,12,"gray25").
  kcolour_table( green,      green, pale,  mid,   10,9,"gray50").
% kcolour_table( green,      green, pale,  high,  10,9,"").
% kcolour_table( cyan,       cyan,  dark,  low,
% kcolour_table( cyan,       cyan,  dark,  mid
  kcolour_table( cyan,       cyan,  mid,   low,   3,12,"gray75").
  kcolour_table( cyan,       cyan,  mid,   mid,   3,1,"").
% kcolour_table( cyan,       cyan,  mid,   high,
  kcolour_table( cyan,       cyan,  light, low,   3,12,"gray25").
  kcolour_table( cyan,       cyan,  light, mid,   3,9,"gray50").
  kcolour_table( cyan,        cyan,  light, high,     11,1,"").         % lightcyan
% kcolour_table( cyan,        cyan,  pale,  high,     11,12,"").
  kcolour_table( cyan,        cyan,  pale,  mid,      11,12,"gray50").
  kcolour_table( cyan,        cyan,  pale,  high,     11,12,"gray75").
% kcolour_table( red,         red,   dark,  low,      4,1,"").          % red
% kcolour_table( red,         red,   dark,  mid,
% kcolour_table( red,         red,   mid,   low,
  kcolour_table( red,         red,   mid,   mid,      4,12,"gray75").
  kcolour_table( red,         red,   mid,   high,     4,1,"").          % red 36
  kcolour_table( red,         red,   light, low,      4,12,"gray50").
  kcolour_table( red,         red,   light, mid,      4,12,"gray50").
  kcolour_table( red,         red,   light, high,     4,12,"gray25").
  kcolour_table( red,         red,   pale,  mid,      4,11,"").
% kcolour_table( red,         red,   pale,  high,     4,11,"").
  kcolour_table( magenta,     magenta,dark,  low,     5,1,"gray75").
  kcolour_table( magenta,     magenta,dark,  mid,     5,1,"").          % magenta
% kcolour_table( magenta,     magenta,dark,  high,    5,1,"").
  kcolour_table( magenta,     magenta,mid,   low,     5,12,"gray50").
  kcolour_table( magenta,     magenta,mid,   mid,     5,12,"gray75").
  kcolour_table( magenta,     magenta,mid,   high,    13,1,"").         % lightmagenta
  kcolour_table( magenta,     magenta,light, low,     5,12,"gray25").
  kcolour_table( magenta,     magenta,light, mid,     13,12,"gray75").
  kcolour_table( magenta,     magenta,light, high,    13,1,"").
  kcolour_table( magenta,     magenta,pale,  low,     13,12,"gray25").
  kcolour_table( magenta,     magenta,pale,  mid,     13,12,"gray50").
% kcolour_table( magenta,     magenta,pale,  high,    13,9,"").
% kcolour_table( purple,      purple,dark,  low,      5,1,"").
% kcolour_table( purple,      purple,dark,  mid,      5,1,"").          % magenta
% kcolour_table( purple,      purple,mid,   low,      5,9,"").
% kcolour_table( purple,      purple,mid,   mid,      5,9,"").
% kcolour_table( brown,       brown, dark,  low,
% kcolour_table( brown,       brown, dark,  mid,
  kcolour_table( brown,       brown, mid,   low,      6,12,"gray75").
  kcolour_table( brown,       brown, mid,   mid,      6,1,"").          % brown
% kcolour_table( brown,       brown, mid,   high,
  kcolour_table( brown,       brown, light, low,      6,12,"gray25").
  kcolour_table( brown,       brown, light, mid,      6,12,"gray50").
% kcolour_table( brown,       brown, light, high,
  kcolour_table( brown,       brown, pale,  mid,      6,11,"").
% kcolour_table( brown,       brown, pale,  high,
  kcolour_table( lightgray,   grey,  mid,    low,     7,1,"").          % lightgray
  kcolour_table( darkgray,    grey,  dark,   low,     8,1,"").          % darkgrey
  kcolour_table( lightblue,   blue,  light,  mid,     9,1,"").          % lightblue
  kcolour_table( lightgreen,  green, light,  high,    10,1,"").         % lightgreen
  kcolour_table( lightcyan,   cyan,  light,  high,    11,1,"").         % lightcyan
% kcolour_table( lightred,    red,   mid,    high,    12,1,"").         % lightred
  kcolour_table( lightmagenta, magenta,light,high,    13,1,"").         % lightmagenta
% kcolour_table( yellow,      yellow, dark,  low,
% kcolour_table( yellow,      yellow, dark,  mid,
% kcolour_table( yellow,      yellow, mid,   low,
% kcolour_table( yellow,      yellow, mid,   mid,
% kcolour_table( yellow,      yellow, mid,   high,
% kcolour_table( yellow,      yellow, light, low,     14,9,"").
```

```
  kcolour_table( yellow,     yellow, light,  mid,    14,12,"gray75").
  kcolour_table( yellow,     yellow, light,  high,   14,1,"").% yellow
  kcolour_table( yellow,     yellow, pale,   low,    14,12,"gray25").
  kcolour_table( yellow,     yellow, pale,   mid,    14,12,"gray50").
  kcolour_table( yellow,     yellow, pale,   high,   14,12,"gray75").
  kcolour_table( white,      white,  pale,   high,   0,1,"").          % white
  kcolour_table( white,      white,  light,  high,   0,1,"").
  kcolour_table( grey,       grey,   dark,   low,    8,1,"").          % darkgrey
% kcolour_table( grey,       grey,   dark,   mid
  kcolour_table( grey,       grey,   mid,    low,    8,12,"gray50").
% kcolour_table( grey,       grey,   mid,    mid,
% kcolour_table( grey,       grey,   mid,    high,
  kcolour_table( grey,       grey,   light,  low,    7,1,"").          %lightgray
% kcolour_table( grey,       grey,   light,  mid,
% kcolour_table( grey,       grey,   light,  high,
  kcolour_table( grey,       grey,   pale,   low,    7,12,"gray50").
% kcolour_table( grey,       grey,   pale,   high,
% kcolour_table(,dark, low,
% kcolour_table(,dark, mid,
% kcolour_table( orange,     orange, mid,    low,    12,1,"").
  kcolour_table( orange,     orange, mid,    mid,    12,1,""). % orange 12
% kcolour_table( orange,     orange, mid,    high,
  kcolour_table( orange,     orange, light,  low,    12,9,"").
  kcolour_table( orange,     orange, light,  mid,    12,12,"gray75").
% kcolour_table( orange,     orange, light,  high,
  kcolour_table( orange,     orange, pale,   low,    12,12,"gray25").
  kcolour_table( orange,     orange, pale,   mid,    12,12,"gray50").
% kcolour_table( orange,     orange, pale,   high,


  kpattern("",[]).
  kpattern( t1, [$00, $80, $00, $08, $00, $80, $00, $08]).
  kpattern( t2, [$00, $88, $00, $22, $00, $88, $00, $22]).
  kpattern( t3, [$00, $AA, $00, $55, $00, $AA, $00, $55]).
  kpattern( t4, [$AA, $11, $AA, $44, $AA, $11, $AA, $44]).
  kpattern( t5, [$AA, $55, $AA, $55, $AA, $55, $AA, $55]).
  kpattern( t6, [$AA, $FF, $55, $FF, $AA, $FF, $55, $FF]).
  kpattern( t7, [$FF, $BB, $FF, $EE, $FF, $BB, $FF, $EE]).
  kpattern( t8, [$FF, $7F, $FF, $F7, $FF, $7F, $FF, $F7]).
  kpattern( solid, [$FF, $FF, $FF, $FF, $FF, $FF, $FF, $FF]).

  kpattern( gray06, [$00, $80, $00, $08, $00, $80, $00, $08]).
  kpattern( gray12, [$00, $88, $00, $22, $00, $88, $00, $22]).
  kpattern( gray25, [$88, $22, $88, $22, $88, $22, $88, $22]).
  kpattern( gray37, [$AA, $11, $AA, $44, $AA, $11, $AA, $44]).
  kpattern( gray50, [$AA, $55, $AA, $55, $AA, $55, $AA, $55]).
  kpattern( gray75, [$AA, $FF, $55, $FF, $AA, $FF, $55, $FF]).
  kpattern( gray88, [$FF, $BB, $FF, $EE, $FF, $BB, $FF, $EE]).
  kpattern( gray94, [$FF, $7F, $FF, $F7, $FF, $7F, $FF, $F7]).
  kpattern( gray100, [$FF, $FF, $FF, $FF, $FF, $FF, $FF, $FF]).
  kpattern( finehatch75, [$AA, $FF, $AA, $FF, $AA, $FF, $AA, $FF]).
  kpattern( pattern37, [$88, $11, $88, $33, $88, $11, $88, $33]).
  kpattern( pattern75, [$EE, $BB, $DD, $77, $EE, $BB, $DD, $77]).
  kpattern( pattern25,  [$00, $AA, $00, $55, $00, $AA, $00, $55]).
  kpattern( diag25, [$88, $44, $22, $11,$88, $44, $22, $11]).


  kline_colour( black,        black, dark,  low,   15).   % black
  kline_colour( black,        black, "",    "",    15).
  kline_colour( dark_blue,    blue,  dark,  mid,   1).    % blue
  kline_colour( blue,         blue,  mid,   high,  9).    % lightblue
  kline_colour( blue,         blue,  "",    "",    1).
  kline_colour( light_blue,   blue,  light, high,  11).   % lightcyan
  kline_colour( green,        green, mid,   low,   2).    % green
  kline_colour( green,        green, "",    "",    2).
  kline_colour( light_green,  green, light, high,  10).   % lightgreen
  kline_colour( dark_cyan,    cyan,  mid,   low,   3).    % cyan
  kline_colour( cyan,         cyan,  light, high,  11).   % lightcyan
  kline_colour( cyan,         cyan,  "",    "",    11).
  kline_colour( red,          red,   mid,   mid,   4).    % red 36
  kline_colour( red,          red,   "",    "",    4).
  kline_colour( purple,       purple, dark, mid,   5).    % magenta
  kline_colour( purple,       magenta,dark,mid,    5).
  kline_colour( purple,       purple,"",    "",    5).
  kline_colour( magenta,      magenta,mid, high,   13).   % lightmagenta
  kline_colour( magenta,      magenta,"",   "",    13).
  kline_colour( brown,        brown, mid,   low,   6).    % brown
  kline_colour( brown,        brown, "",    "",    6).
  kline_colour( orange,       orange, mid,  mid,   12).   % orange
  kline_colour( orange,       orange, "",   "",    12).
  kline_colour( yellow,       yellow, light, high, 14).   % yellow
  kline_colour( yellow,       yellow, "",   "",    14).
  kline_colour( light_grey,   grey,  light, low,   7).    % lightgray
  kline_colour( grey,         grey,  "",    "",    7).
  kline_colour( dark_grey,    grey,  dark,  low,   8).    % darkgray
  kline_colour( dark_grey,    grey,  mid,   low,   8).
  kline_colour( white,        white, pale,  low,   0).    % white
  kline_colour( white,        white, light, low,   0).
  kline_colour( white,        white, "",    "",    0).
%   kline_colour(

  kline_form( continuous,     "",    0,     0).
  kline_form( dotted,         "",    1,     0).
  kline_form( dashed,         "",    3,     0).
  kline_form( chain,          "",    2,     0).
  kline_form( cased,          "",    0,     0).
  kline_form( complex,        "",    4,     0).

  kline_width( fine,    1).
  kline_width( thin,    1).
  kline_width( medium, 3).
  kline_width( thick,   3).
  kline_width( wide,    3).

  kpoint_form( geometric,     pixel,        0).
  kpoint_form( geometric,     circle,       1).    % unfilled
kpoint_form( geometric,    dot,          2).
kpoint_form( geometric,    box,          3).     % unfilled
kpoint_form( geometric,    square,       4).     % filled
kpoint_form( geometric,    plus,         5).
kpoint_form( geometric,    cross,        6).

kpoint_size( pixel,        0).
kpoint_size( v_small,      2).
kpoint_size( small,        3).
kpoint_size( medium,       4).
kpoint_size( large,        5).
kpoint_size( v_large,      6).
kpoint_size( grad0,        1).
kpoint_size( grad1,        3).
kpoint_size( grad2,        6).
kpoint_size( grad3,        9).
kpoint_size( grad4,        12).
kpoint_size( grad5,        15).
kpoint_size( grad6,        18).
kpoint_size( grad7,        21).
kpoint_size( grad8,        24).
kpoint_size( grad9,        27).

klookup(cased, main_highways,red, yellow, red).
klookup(cased, highways,  red, red, red).
klookup(cased, other_roads, red,  red, red).              % thin
klookup(cased, main_highways,yellow, red, yellow).
klookup(cased, highways,  yellow, yellow, yellow).
klookup(cased, other_roads, yellow, yellow, yellow).      % thin
klookup(cased, main_highways,grey, red, grey).
klookup(cased, highways,  grey,   yellow, grey).
klookup(cased, other_roads, grey, grey, grey).            % thin


GOAL
 save("lookup.kba",lookup).
```

APPENDIX D

Sample map frames and output

All the maps illustrated here are the direct output from the system using the values in the accompanying frame

1. Nigeria topographic
   The topographic map produced with default parameters.
2. Nigeria topographic
   The only difference in input to the above is the selection of 'specialist user' and 'anlysis' for map_purpose. As can be seen, more detail is shown.
3. S.E. Nigeria topographic
   Using the same input as example 1, except for the location. The resulting larger scale automatically includes more detail.
4. Nigeria population density
5. Nigeria total population
6. Nigeria rural population

1. Nigeria topographic

The topographic map produced with default parameters.

```
fmap_date(dated(1995,6,15))
fmap_author("df")
fmap_title("Nigeria Topographic")
fmap_type("topographic")
fmap_purpose("overview")
fmap_user("general")
foutput_media("screen")
flevel_of_detail(2)
fscale(7500000)
fformat(18,16.5)
flat_long(2.5,15,3.5,14)
flimits(277.98731656,388.9402453,1667.9238994,1541.2843966)
fselect_index(10)
fbase_info_list(["Coastline","Seas","Major Rivers","Lakes","Lake_fill","International
Boundaries","Capitals","Main Highways","Main Relief"])
fbase_info("Main Relief",10)
fbase_info("Main Highways",10)
fbase_info("Capitals",10)
fbase_info("International Boundaries",10)
fbase_info("Lakes",10)
fbase_info("Major Rivers",10)
fbase_info("Coastline",10)
ftheme_info_list([])
frepresentation("Coastline","boundaries - one level")
frepresentation("Seas","isolated areas")
frepresentation("Rivers","network - branching")
frepresentation("Lakes","boundaries - one level")
frepresentation("Lake_fill","isolated areas")
frepresentation("Administrative Boundaries","boundaries - hierarchy")
frepresentation("Settlements","ranked points")
frepresentation("Roads","network - link & node")
frepresentation("Relief","hypsometric layers")
fsymbolism("Coastline","Coastline",symbolspec("line","blue","dark","low","continuou
s","",0,"fine",201))
fsymbolism("Seas","Seas",symbolspec("area","cyan","light","low","solid","",0,"",158))
fsymbolism("Rivers","Major
Rivers",symbolspec("line","blue","dark","mid","continuous","",0,"fine",190))
fsymbolism("Administrative Boundaries","International
Boundaries",symbolspec("line","grey","dark","low","chain","chain",0,"thick",210))
fsymbolism("Lakes","Lakes",symbolspec("line","blue","dark","mid","continuous","",0,
"fine",202))
fsymbolism("Lake_fill","Lake_fill",symbolspec("area","cyan","light","low","solid","",0,"
",191))
fsymbolism("Settlements","Capitals",symbolspec("point","magenta","dark","mid","ge
ometric","square",0,"medium",251))
fsymbolism("Roads","Main
Highways",symbolspec("line","red","mid","mid","cased","main_highways",0,"thick",2
23))
fsymbolism("Relief","0-
500",symbolspec("area","green","mid","low","solid","",0,"",141))
```

```
fsymbolism("Relief","500-
2000",symbolspec("area","brown","mid","low","solid","",0,"",141))
fsymbolism("Relief","over
2000",symbolspec("area","brown","mid","mid","solid","",0,"",141))
```

2. Nigeria topographic

The only difference in input to the previous example is the selection of 'specialist user' and 'anlysis' for map_purpose. As can be seen, more detail is shown.

```
fmap_date(dated(1995,6,15))
fmap_author("df")
fmap_title("Nigeria Topographic")
fmap_type("topographic")
fmap_purpose("analysis")
fmap_user("specialist")
foutput_media("screen")
flevel_of_detail(8)
fscale(7500000)
fformat(18,16.5)
flat_long(2.5,15,3.5,14)
flimits(277.98731656,388.9402453,1667.9238994,1541.2843966)
fselect_index(7)
fbase_info_list(["Coastline","Seas","Large Rivers","Major
Rivers","Lakes","Lake_fill","State Boundaries","International Boundaries","Major
Towns","Capitals","Highways","Main Highways","Railways","Main Relief"])
fbase_info("Main Relief",10)
fbase_info("Railways",8)
fbase_info("Highways",8)
fbase_info("Main Highways",10)
fbase_info("Major Towns",8)
fbase_info("Capitals",10)
fbase_info("State Boundaries",8)
fbase_info("International Boundaries",10)
fbase_info("Lakes",10)
fbase_info("Large Rivers",8)
fbase_info("Major Rivers",10)
fbase_info("Coastline",10)
ftheme_info_list([])
frepresentation("Coastline","boundaries - one level")
frepresentation("Seas","isolated areas")
frepresentation("Rivers","network - branching")
frepresentation("Lakes","boundaries - one level")
frepresentation("Lake_fill","isolated areas")
frepresentation("Administrative Boundaries","boundaries - hierarchy")
frepresentation("Settlements","ranked points")
frepresentation("Roads","network - link & node")
frepresentation("Railways","network - link & node")
frepresentation("Relief","hypsometric layers")
fsymbolism("Coastline","Coastline",symbolspec("line","blue","dark","low","continuou
s","",0,"fine",201))
fsymbolism("Seas","Seas",symbolspec("area","cyan","light","low","solid","",0,"",158))
fsymbolism("Rivers","Major
Rivers",symbolspec("line","blue","dark","mid","continuous","",0,"fine",190))
fsymbolism("Rivers","Large
Rivers",symbolspec("line","blue","dark","mid","continuous","",0,"fine",190))
```

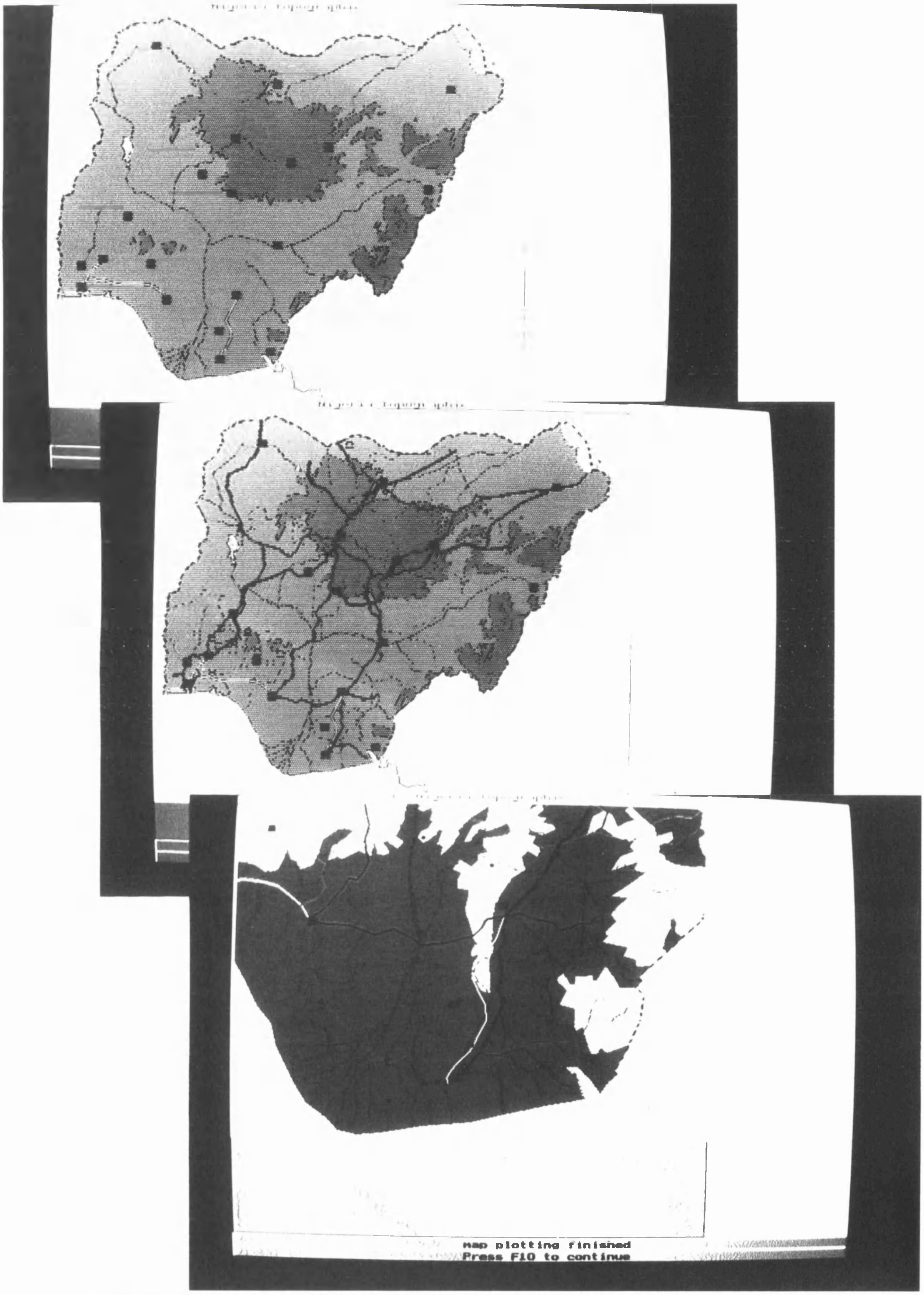
```
fsymbolism("Administrative Boundaries","International
Boundaries",symbolspec("line","grey","dark","low","chain","chain",0,"thick",210))
fsymbolism("Administrative Boundaries","State
Boundaries",symbolspec("line","black","dark","low","chain","chain",0,"fine",210))
fsymbolism("Railways","Railways",symbolspec("line","grey","dark","low","continuous
","",0,"medium",221))
fsymbolism("Lakes","Lakes",symbolspec("line","blue","dark","mid","continuous","",0,
"fine",202))
fsymbolism("Lake_fill","Lake_fill",symbolspec("area","cyan","light","low","solid","",0,"
",191))
fsymbolism("Settlements","Capitals",symbolspec("point","magenta","dark","mid","ge
ometric","square",0,"medium",251))
fsymbolism("Settlements","Major
Towns",symbolspec("point","magenta","dark","mid","geometric","box",0,"small",251)
)
fsymbolism("Roads","Main
Highways",symbolspec("line","red","mid","mid","cased","main_highways",0,"thick",2
23))
fsymbolism("Roads","Highways",symbolspec("line","red","mid","mid","cased","highw
ays",0,"thick",223))
fsymbolism("Relief","0-
500",symbolspec("area","green","mid","low","solid","",0,"",141))
fsymbolism("Relief","500-
2000",symbolspec("area","brown","mid","low","solid","",0,"",141))
fsymbolism("Relief","over
2000",symbolspec("area","brown","mid","mid","solid","",0,"",141))
```

3. S.E. Nigeria topographic

Using the same input as example 1, except for the location. The resulting larger scale automatically includes more detail.

```
fmap_date(dated(1995,6,15))
fmap_author("df")
fmap_title("S.E. Nigeria Topographic")
fmap_type("topographic")
fmap_purpose("overview")
fmap_user("author")
foutput_media("screen")
flevel_of_detail(4)
fscale(3000000)
fformat(18,16.5)
flat_long(4.9,9.5,4.2,7.5)
flimits(544.85514046,466.60055381,1056.3518029,831.58237048)
fselect_index(5)
fbase_info_list(["Coastline","Seas","Other Rivers","Large Rivers","Major
Rivers","Lakes","Lake_fill","Tertiary Boundaries","State Boundaries","International
Boundaries","Minor Towns","Major Towns","Capitals","Urban
Areas","Highways","Main Highways","Railways","Minor Relief","Main Relief"])
fbase_info("Minor Relief",6)
fbase_info("Main Relief",10)
fbase_info("Railways",8)
fbase_info("Highways",8)
fbase_info("Main Highways",10)
fbase_info("Urban Areas",6)
fbase_info("Minor Towns",6)
fbase_info("Major Towns",8)
fbase_info("Capitals",10)
fbase_info("Tertiary Boundaries",6)
fbase_info("State Boundaries",8)
fbase_info("International Boundaries",10)
fbase_info("Lakes",10)
fbase_info("Other Rivers",6)
fbase_info("Large Rivers",8)
fbase_info("Major Rivers",10)
fbase_info("Coastline",10)
ftheme_info_list([])
frepresentation("Coastline","boundaries - one level")
frepresentation("Seas","isolated areas")
frepresentation("Rivers","network - branching")
frepresentation("Lakes","boundaries - one level")
frepresentation("Lake_fill","isolated areas")
frepresentation("Administrative Boundaries","boundaries - hierarchy")
frepresentation("Settlements","ranked points")
frepresentation("Urban Areas","isolated areas")
frepresentation("Roads","network - link & node")
frepresentation("Railways","network - link & node")
frepresentation("Relief","hypsometric layers")
```

```
fsymbolism("Coastline","Coastline",symbolspec("line","blue","dark","low","continuou
s","",0,"fine",201))
fsymbolism("Seas","Seas",symbolspec("area","cyan","light","low","solid","",0,"",158))
fsymbolism("Rivers","Major
Rivers",symbolspec("line","blue","dark","mid","continuous","",0,"fine",190))
fsymbolism("Rivers","Large
Rivers",symbolspec("line","blue","dark","mid","continuous","",0,"fine",190))
fsymbolism("Rivers","Other
Rivers",symbolspec("line","blue","dark","mid","continuous","",0,"fine",190))
fsymbolism("Administrative Boundaries","International
Boundaries",symbolspec("line","grey","dark","low","chain","chain",0,"thick",210))
fsymbolism("Administrative Boundaries","State
Boundaries",symbolspec("line","black","dark","low","chain","chain",0,"fine",210))
fsymbolism("Administrative Boundaries","Tertiary
Boundaries",symbolspec("line","grey","dark","low","dotted","dotted",0,"fine",210))
fsymbolism("Railways","Railways",symbolspec("line","grey","dark","low","continuous
","",0,"medium",221))
fsymbolism("Lakes","Lakes",symbolspec("line","blue","dark","mid","continuous","",0,
"fine",202))
fsymbolism("Lake_fill","Lake_fill",symbolspec("area","cyan","light","low","solid","",0,"
",191))
fsymbolism("Settlements","Capitals",symbolspec("point","magenta","dark","mid","ge
ometric","square",0,"medium",251))
fsymbolism("Settlements","Major
Towns",symbolspec("point","magenta","dark","mid","geometric","box",0,"small",251)
)
fsymbolism("Settlements","Minor
Towns",symbolspec("point","magenta","dark","mid","geometric","dot",0,"v_small",25
1))
fsymbolism("Urban Areas","Urban
Areas",symbolspec("type","hue","light","sat","code","form",0,"dim",152))
fsymbolism("Roads","Main
Highways",symbolspec("line","red","mid","mid","cased","main_highways",0,"thick",2
23))
fsymbolism("Roads","Highways",symbolspec("line","red","mid","mid","cased","highw
ays",0,"thick",223))
fsymbolism("Relief","0-
200",symbolspec("area","green","mid","mid","solid","",0,"",141))
fsymbolism("Relief","200-
500",symbolspec("area","green","light","mid","solid","",0,"",141))
fsymbolism("Relief","500-
1000",symbolspec("area","orange","light","mid","solid","",0,"",141))
fsymbolism("Relief","1000-
2000",symbolspec("area","brown","mid","low","solid","",0,"",141))
fsymbolism("Relief","over
2000",symbolspec("area","brown","mid","mid","solid","",0,"",141))
```



Map plotting finished
Press F10 to continue

Figures D.1, D.2, D.3

Overview topographic map, analysis topographic map, larger scale topographic map.

4. Nigeria population density

```
fmap_date(dated(1995,6,15))
fmap_author("df")
fmap_title("Nigeria Population Density")
fmap_type("population")
fmap_purpose("overview")
fmap_user("author")
foutput_media("screen")
flevel_of_detail(4)
fscale(7500000)
fformat(18,16.5)
flat_long(2.5,15,3.5,14)
flimits(277.98731656,388.9402453,1667.9238994,1541.2843966)
fselect_index(9)
fbase_info_list(["Coastline","Seas","State Boundaries","International
Boundaries","Major Towns","Capitals","Main Highways"])
fbase_info("Main Highways",9)
fbase_info("Major Towns",9)
fbase_info("Capitals",10)
fbase_info("State Boundaries",9)
fbase_info("International Boundaries",10)
fbase_info("Coastline",10)
ftheme_info_list(["Population density"])
frepresentation("Population density","graded series - unipolar")
frepresentation("Coastline","boundaries - one level")
frepresentation("Seas","isolated areas")
frepresentation("Administrative Boundaries","boundaries - hierarchy")
frepresentation("Settlements","ranked points")
frepresentation("Roads","network - link & node")
fclass_intervals("Population density",[22,60.55,105.6,164.4,228.45,370])
fsymbolism("Coastline","Coastline",symbolspec("line","blue","dark","low","continuou
s","",0,"fine",201))
fsymbolism("Seas","Seas",symbolspec("area","cyan","light","low","solid","",0,"",158))
fsymbolism("Administrative Boundaries","International
Boundaries",symbolspec("line","grey","dark","low","chain","chain",0,"thick",210))
fsymbolism("Administrative Boundaries","State
Boundaries",symbolspec("line","black","dark","low","chain","chain",0,"fine",210))
fsymbolism("Population
density","class1",symbolspec("class1","green","dark","high","tint","t2",0,"",121))
fsymbolism("Population
density","class2",symbolspec("class2","green","dark","high","tint","t3",0,"",121))
fsymbolism("Population
density","class3",symbolspec("class3","green","dark","high","tint","t5",0,"",121))
fsymbolism("Population
density","class4",symbolspec("class4","green","dark","high","tint","t7",0,"",121))
fsymbolism("Population
density","class5",symbolspec("class5","green","dark","high","tint","solid",0,"",121))
fsymbolism("Settlements","Capitals",symbolspec("point","magenta","dark","mid","ge
ometric","square",0,"medium",251))
fsymbolism("Settlements","Major
Towns",symbolspec("point","magenta","dark","mid","geometric","box",0,"small",251)
)
```

```
fsymbolism("Roads","Main
Highways",symbolspec("line","red","mid","mid","cased","main_highways",0,"thick",2
23))
```

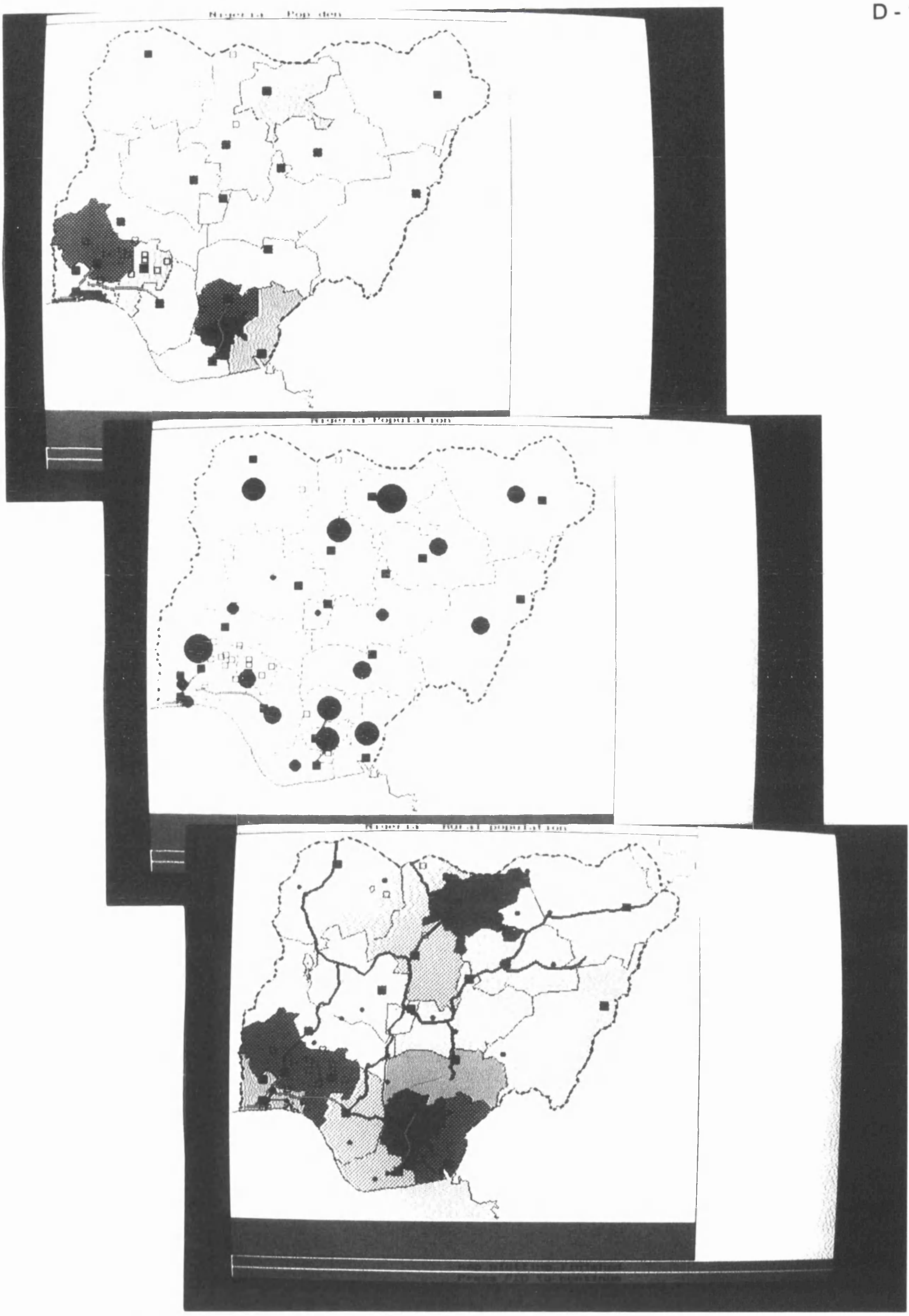
5. Nigeria total population

```
fmap_date(dated(1995,6,15))
fmap_author("")
fmap_title("Nigeria population")
fmap_type("population")
fmap_purpose("overview")
fmap_user("general")
foutput_media("screen")
flevel_of_detail(2)
fscale(7500000)
fformat(18,16.5)
flat_long(2.5,15,3.5,14)
flimits(277.98731656,388.9402453,1667.9238994,1541.2843966)
fselect_index(10)
fbase_info_list(["Coastline","Seas","International Boundaries","Capitals"])
fbase_info("Capitals",10)
fbase_info("International Boundaries",10)
fbase_info("Coastline",10)
ftheme_info_list(["Total population"])
frepresentation("Total population","proportional - classed")
frepresentation("Coastline","boundaries - one level")
frepresentation("Seas","isolated areas")
frepresentation("Administrative Boundaries","boundaries - hierarchy")
frepresentation("Settlements","ranked points")
fclass_intervals("Total
population",[200000,1320000,2440000,3560000,4680000,5800000])
fsymbolism("Coastline","Coastline",symbolspec("line","blue","dark","low","continuou
s","",0,"fine",201))
fsymbolism("Seas","Seas",symbolspec("area","cyan","light","low","solid","",0,"",158))
fsymbolism("Administrative Boundaries","International
Boundaries",symbolspec("line","grey","dark","low","chain","chain",0,"thick",210))
fsymbolism("Total
population","class1",symbolspec("point","red","mid","high","geometric","dot",0,"grad
1",241))
fsymbolism("Total
population","class2",symbolspec("point","red","mid","high","geometric","dot",0,"grad
2",241))
fsymbolism("Total
population","class3",symbolspec("point","red","mid","high","geometric","dot",0,"grad
3",241))
fsymbolism("Total
population","class4",symbolspec("point","red","mid","high","geometric","dot",0,"grad
4",241))
fsymbolism("Total
population","class5",symbolspec("point","red","mid","high","geometric","dot",0,"grad
5",241))
fsymbolism("Settlements","Capitals",symbolspec("point","magenta","dark","mid","ge
ometric","square",0,"medium",251))
```

6. Nigeria rural population

```
fmap_date(dated(1995,6,15))
fmap_author("df")
fmap_title("Nigeria - rural population")
fmap_type("rural")
fmap_purpose("analysis")
fmap_user("author")
foutput_media("screen")
flevel_of_detail(8)
fscale(7500000)
fformat(18,16.5)
flat_long(2.5,15,3.5,14)
flimits(277.98731656,388.9402453,1667.9238994,1541.2843966)
fselect_index(7)
fbase_info_list(["Coastline","Seas","Lakes","Lake_fill","Tertiary Boundaries","State
Boundaries","International Boundaries","Minor Towns","Major
Towns","Capitals","Highways","Main Highways"])
fbase_info("Highways",7)
fbase_info("Main Highways",9)
fbase_info("Minor Towns",8)
fbase_info("Major Towns",9)
fbase_info("Capitals",10)
fbase_info("Tertiary Boundaries",8)
fbase_info("State Boundaries",9)
fbase_info("International Boundaries",10)
fbase_info("Lakes",8)
fbase_info("Coastline",10)
ftheme_info_list(["Rural population"])
frepresentation("Rural population","graded series - unipolar")
frepresentation("Coastline","boundaries - one level")
frepresentation("Seas","isolated areas")
frepresentation("Lakes","boundaries - one level")
frepresentation("Lake_fill","isolated areas")
frepresentation("Administrative Boundaries","boundaries - hierarchy")
frepresentation("Settlements","ranked points")
frepresentation("Roads","network - link & node")
fclass_intervals("Rural population",[18,21,34,50.5,65,112,290])
fsymbolism("Coastline","Coastline",symbolspec("line","blue","dark","low","continuou
s","",0,"fine",201))
fsymbolism("Seas","Seas",symbolspec("area","cyan","light","low","solid","",0,"",158))
fsymbolism("Administrative Boundaries","International
Boundaries",symbolspec("line","grey","dark","low","chain","chain",0,"thick",210))
fsymbolism("Administrative Boundaries","State
Boundaries",symbolspec("line","black","dark","low","chain","chain",0,"fine",210))
fsymbolism("Administrative Boundaries","Tertiary
Boundaries",symbolspec("line","grey","dark","low","dotted","dotted",0,"fine",210))
fsymbolism("Rural
population","class1",symbolspec("class1","green","dark","high","tint","t2",0,"",121))
fsymbolism("Rural
population","class2",symbolspec("class2","green","dark","high","tint","t3",0,"",121))
fsymbolism("Rural
population","class3",symbolspec("class3","green","dark","high","tint","t4",0,"",121))
```

```
fsymbolism("Rural
population","class4",symbolspec("class4","green","dark","high","tint","t5",0,"",121))
fsymbolism("Rural
population","class5",symbolspec("class5","green","dark","high","tint","t7",0,"",121))
fsymbolism("Rural
population","class6",symbolspec("class6","green","dark","high","tint","solid",0,"",121)
)
fsymbolism("Lakes","Lakes",symbolspec("line","blue","dark","low","continuous","",0,"
fine",202))
fsymbolism("Lake_fill","Lake_fill",symbolspec("area","cyan","light","low","solid","",0,"
",191))
fsymbolism("Settlements","Capitals",symbolspec("point","magenta","dark","mid","ge
ometric","square",0,"medium",251))
fsymbolism("Settlements","Major
Towns",symbolspec("point","magenta","dark","mid","geometric","box",0,"small",251)
)
fsymbolism("Settlements","Minor
Towns",symbolspec("point","magenta","dark","mid","geometric","dot",0,"v_small",25
1))
fsymbolism("Roads","Main
Highways",symbolspec("line","red","mid","mid","cased","main_highways",0,"thick",2
23))
fsymbolism("Roads","Highways",symbolspec("line","red","mid","mid","cased","highw
ays",0,"thick",223))
```



Figures D.4, D.5, D.6

Population density map, Total population map, Rural population density map.

APPENDIX E

File Formats

1. Polygons

polygon

seq_no	name	npts	area	xcent	ycent	[coords]	
integer	string	integer	real	integer	integer	integerlist	
				long	lat	[long,lat,]	
			sq.units	ddd.dd	ddd.dd		
1	"Bauchi"	68	14699	743	234	[664,324, 667,642, ... 664,324]	

notes: coords MUST close back on start point

e.g.

polygon(1,"Bauchi",68,14669,743,634,[664,324,667,642, . . . ,664,324]).

2. Lines

chain

seq_no	feat_code	right_pol or name	left -pol	npts	st_node	end_node	[coords]
integer	string	string	string	integer	integer	integer	integerlist
							[long,lat,]
1	"state_bnd"	"Bauchi"	"Bendel"	13	33	34	[664,324, 667,642, ... 483,247]

e.g.

complete specification:

chain(1,"state_bnd","Bauchi",Bendel",23,33,34,[664,324,667,642, . . . ,483,247]).

area chain:

chain(1,"","Bauchi",Bendel",23,0,0,[664,324,667,642, . . . ,483,247]).

network chain:

chain(1,"major_road","","",23,33,34,[664,324,667,642, . . . ,483,247]).

other lines:

chain(1,"fault_line","",",23,0,0,[664,324,667,642, . . . ,483,247]).

3. Points

node

seq_no	**feat_code**	**xcoord**	**ycoord**	**name**	**value**		
integer	string	integer	integer	string	real		
		long	lat				
		ddd.dd	ddd.dd				
1	"State"	532	1310	"Sokoto"	148000		

e.g. - settlement - f_c = National, State,Major,Minor. value = population
node(1,"State",532,1310,"Sokoto",148000).

4. Attributes (point attributes or polygon attributes)

feature_class

seq_no	**name**	**xcoord**	**ycoord**	**value_1**	**value_2**	**. . .**	**value_n**
integer	string	integer	integer	real			
		long	lat				
		ddd.dd	ddd.dd				
1	"Sokoto"	532	1310	148000			

notes: each feature class will have its own declaration depending on number & type of attributes
attributes may be string, real, or integer.
for polygons to be shaded names must exactly match those in polygon file.

e.g.
v_1 = population, v_2 = males, v_3 = females, v_4 = unemployed, etc.
state(1,Sokoto,532,1310,148000,70000,78000,5000).

APPENDIX F

Metadata for database information

(see Chapter 7, page 170)

Feature class metadata

```
kmeta_data("Coastline",dated(0,0,1993),"Collins Atlas",
        "linear","line","identical","tangible","Coastline","","","")
kmeta_data("Administrative Boundaries",dated(0,0,1993),"Collins Atlas",
        "linear","line","hierarchical_feature_coded","tangible",
        "Administrative Boundaries","","","")
kmeta_data("Administrative areas",dated(0,0,1993),"Collins Atlas",
        "specific_area","polygon","identical","tangible","Administrative areas","","","")
kmeta_data("Seas",dated(0,0,1993),"Collins Atlas",
        "specific_area","polygon","identical","tangible","Seas","","","")
kmeta_data("Settlements",dated(0,0,93),"Collins Atlas",
        "discrete","point","hierarchical_feature_coded","tangible","Settlements","","",
        "")
kmeta_data("Urban population",dated(0,0,1993),"Collins Atlas",
        "discrete","point","ratio_abs","tangible",
        "Settlements","","Settlements","population")
kmeta_data("Total population",dated(0,0,94),"World Factbook",
        "discrete","polygon","ratio_abs","tangible",
        "Administrative areas","","State population","total")
kmeta_data("Population density",dated(0,0,94),"World Factbook",
        "discrete","polygon","ratio_den","tangible",
        "Administrative areas","","State population","density")
kmeta_data("Rural population",dated(0,0,94),"World Factbook",
        "discrete","polygon","ratio_abs","tangible",
        "Administrative areas","","State population","rural density")
kmeta_data("Railways",dated(0,0,1993),"Collins Atlas",
        "linear","line","hierarchical_feature_coded","tangible",
        "Communication routes","","","")
kmeta_data("Roads",dated(0,0,1993),"Collins Atlas",
        "linear","line","hierarchical_feature_coded","tangible",
        "Communication routes","","","")
kmeta_data("Rivers",dated(0,0,1993),"Collins Atlas",
        "linear","line","hierarchical_feature_coded","tangible","Rivers","","","")
```

```
kmeta_data("Lakes",dated(0,0,1993),"Collins Atlas",
        "linear","line","identical","tangible","Lakes","","","")
kmeta_data("Lake_fill",dated(0,0,1993),"Collins Atlas",
        "specific_area","polygon","identical","tangible","Lake_fill","","","")
kmeta_data("Contours",dated(0,0,1993),"Collins Atlas",
        "surface","line","identical","conceptual","Contours","","","")
kmeta_data("Relief",dated(0,0,1993),"Collins Atlas",
        "surface","area","ratio","conceptual",
        "Relief","c:\\prolog33\\map\\nigeria\\nigrel.lut","","")
```

Coordinate file meta data

```
kcoord_file("Coastline","line",dated(0,0,1993),[2,3,15,15],5000000,
        "Collins Atlas","sphere","degrees",0.01,
        "c:\\prolog33\\map\\nigeria\\nigcoast.arc","")
kcoord_file("Administrative Boundaries","line",dated(0,0,1993),[2,3,15,15],5000000,
        "Collins Atlas","sphere","degrees",0.01,
        "c:\\prolog33\\map\\nigeria\\adminbnd.arc","")
kcoord_file("Administrative areas","polygon",dated(0,0,1993),[2,3,15,15],5000000,
        "Collins Atlas","sphere","degrees",0.01,
        "c:\\prolog33\\map\\nigeria\\nigstate.pol","")
kcoord_file("Seas","polygon",dated(0,0,1993),[2,3,15,15],5000000,
        "Collins Atlas","sphere","degrees",0.01,
        "c:\\prolog33\\map\\nigeria\\nigsea.pol","")
kcoord_file("Settlements","point",dated(0,0,1993),[2,3,15,15],5000000,
        "Collins Atlas","sphere","degrees",0.01,
        "c:\\prolog33\\map\\nigeria\\nigtowns.nod","")
kcoord_file("Communication routes","line",dated(0,0,1993),[2,3,15,15],5000000,
        "Collins Atlas","sphere","degrees",0.01,
        "c:\\prolog33\\map\\nigeria\\nigroad.arc","")
kcoord_file("Rivers","line",dated(0,0,1993),[2,3,15,15],5000000,
        "Collins Atlas","sphere","degrees",0.01,
        "c:\\prolog33\\map\\nigeria\\nigriver.arc","")
kcoord_file("Lakes","line",dated(0,0,1993),[2,3,15,15],5000000,
        "Collins Atlas","sphere","degrees",0.01,
        "c:\\prolog33\\map\\nigeria\\niglakes.arc","")
kcoord_file("Lake_fill","polygon",dated(0,0,1993),[2,3,15,15],5000000,
        "Collins Atlas","sphere","degrees",0.01,
        "c:\\prolog33\\map\\nigeria\\niglakes.pol","")
```

```
kcoord_file("Contours","line",dated(0,0,1993),[2,3,15,15],5000000,
        "Collins Atlas","sphere","degrees",0.01,
        "c:\\prolog33\\map\\nigeria\\nigcont.arc","")
kcoord_file("Relief","polygon",dated(0,0,1993),[2,3,15,15],5000000,
        "Collins Atlas","sphere","degrees",0.01,
        "c:\\prolog33\\map\\nigeria\\nigrel.pol","")
```

Data file metadata

```
kdata_file("Settlements","coord",dated(0,0,1993),"Collins Atlas",
        ["seq_no","feat_code","xcoord","ycoord","name","population"],
        "C:\\prolog33\\map\\nigeria\\nigtowns.nod","")
kdata_file("State population","data",dated(0,0,1994),"World Factbook",
        ["seq_no","State name","rural density","total","density"],
        "c:\\prolog33\\map\\nigeria\\nigstate.val","")
```

APPENDIX G

Related publications by author

Forrest, D. and Pearson, A.W. (1990) Information Sources in Map Design. In Parry and Perkins (Eds.) **Information Sources in Cartography.** London: Bowker-Saur, .

Forrest, D. (1990) "A model of map design for expert systems applications." *Proceedings, Fourth International Symposium on Spatial Data Handling,* , pp.752-761.

Forrest, D. (1991) "Expert Systems for Cartographic Design and Production." *SUC Bulletin*, Vol.24, No.2, pp. 21-27.

Forrest, D. (1991) "A Classification of Map and Symbol Types and Rules for Their Display for Cartographic Expert Systems Applications." *Proceedings, 15th International Cartographic Conference.* , pp. 450-454.

Forrest, D. (1992) The Development of a Frame Based Cartographic Design Expert System. *Occasional Papers,* No.30. Department of Geography and Topographic Science, University of Glasgow.

Review of: Buttenfield B.P., Mark, D.M. (1991) **Map Generalization: Making Rules for Knowledge Representation,** by In: *Cartographic Journal* Vol. 29, No 2, 1992, pp.187-190.

Forrest, D. (1993) "Expert Systems and Cartographic Design". *The Cartographic Journal.* Vol.30, No.2, pp.143-148.

Forrest, D., Pearson, A.W. (1994) "Somewhere over the rainbow." *Proc. AGI94.* Paper 5.3

CHAPTER ELEVEN

Map design

DAVID FORREST AND ALASTAIR PEARSON

Introduction

Map design is the process of selecting and symbolizing the information to be graphically depicted on the map, together with its layout and the determination of the overall appearance of the map. Map design is quite different from many other types of graphic design because the cartographer is constrained by the spatial positions and geographical relationships of the information being depicted.

Much has been written in recent years about the map user and the map as a communication device, for example, Castner and McGrath (1971), and Taylor (1983). Many perceptual tests have been carried out to try to assess and improve map design (see Chapter 29). There is no 'cookbook' for cartographic design; much is done by experience and intuition, although currently there are attempts to formalize many of the processes to enable computers, and in particular, expert systems, to become involved in map design (Eastman, 1987, Muller, *et al.*, 1988). Despite these moves towards a more formal definition of cartographic design, there is a body of opinion which stresses the 'art' in cartography (Arnheim, 1976; Karssen, 1980; Keates, 1984).

No specific mention will be made of map design for computer assisted cartography and few of the references cited consider design for computer graphic displays, which often requires quite different considerations from the traditional map on paper. The literature on this aspect of design is currently very limited.

Map projections

Transforming the spherical surface of the Earth, to the plane flat surface of the map, is termed map projection. The fact that this process cannot be carried out without distortion and that no single perfect map projection exists, forces the cartographer to make a careful selection of an appropriate projection for a given map design.

A projection provides the framework on which the map is based and determines the extent to which maps retain the power of recording factual information with regard to distance, direction and areal extent (Marschner, 1944). It also has a fundamental impact on its appearance and effectiveness.

There are a great number and variety of map projections employed, each preserving one or several properties and serving some cartographic need better than others (Ray, 1976). Until relatively recently, most of the literature on the subject has been technical, providing descriptions of various projections with their graphical and mathematical construction. However, the work of Baker (1986), Newton (1985), Snyder (1984), and Hutchinson (1982) serve to illustrate the ever growing capacity of the computer to relieve the cartographer from the laborious and often complex task of plotting a new projection to suit the individual demands of an original map. As Robinson (1974) rightly points out, more time can now be spent considering the selection of the most suitable projection.

Hsu (1981), Dent (1985) and Robinson *et al.*, (1984) integrate the selection of map projections into the map design process and move away from the earlier detailed descriptions of constructing map projections and complex derivations of formulae. The importance of this selection process is stressed by Hsu (1981):

> 'An expertly chosen map projection helps to focus and amplify the geographic message of the map. Conversely, a poor choice of projection can easily destroy a map.'

Map projections should be seen as utilitarian devices to be used in combination with all other map elements to effectively convey the message of the map to a given audience (Robinson, 1974). In assessing the relative merits of a projection, Dent (1985) suggests that cartographers should pay particular attention to the projection's principal properties of equivalence, conformality, equidistance and azimuthality (preservation of area, shape, distance and direction respectively). The cartographer should also consider where distortion occurs, the effect the projection has on the appearance of the graticule, the ease with which the projection can be re-centred, and its cost of production. In order to make an effective choice, the purpose of the map and the size and shape of the area to be mapped must also be assessed (Steers, 1965).

Those new to the field may well be bewildered by the terminology, often synonymous, that is used to describe the vast array of projections that exist. In response to this, Maling (1968) gives an extensive list of terms in four languages relating to projections.

Hsu (1981) and Balchin (1954) provide useful analyses of projections, their fundamental characteristics and terminology, with the former illustrating the integration in map design of the choice of projection and the decisions made concerning other cartographic elements. Robinson *et al.*, (1984) is a standard introduction to the subject, providing detailed treatment of patterns of distortion. Maling (1973) offers an authoritative work to those specializing in cartography at degree level. However, by way of an introduction to the subject, Maling (1984) provides a greatly shortened version of his more advanced text and, with the aid of clear illustrations, provides a useful reference to the fundamental concepts of map projections.

Deetz and Adams (1945) is a popular reference though to some extent this has been superseded by Snyder (1987) who, in detail, describes projections used by the United States Geological Survey (USGS) and in addition includes descriptions of several other popular or more useful projections and how they are used. The development of a new projection, designed to permit mapping of the earth continuously from satellite with low distortion, is also included. Snyder (1986) details the projections used by the USGS for large scale quadrangles. Comparisons are made between quadrangles mapped on each of the four projection types used. The articles written by Snyder demonstrate how computerization is underpinning a revival of interest in, and continued development of, map projections.

Steers (1965) and Kellaway (1970) group descriptions of the most frequently used projections according to their shared intermediate surfaces. Both are designed as introductions, giving concise descriptions of the principal properties preserved and the effects on meridians and parallels. Both include graphical and trigonometrical methods of construction and a chapter on the choice of appropriate map projections. McDonnell (1979) conveniently groups the projections according to their shared characteristics and contains some useful worked examples in the text. For similar straightforward and simple explanations reference should also be made to Roblin (1969). Both books avoid complex mathematical explanations as much as possible, though calculations are given for those wishing to gain insight into the simple mathematics behind most of the commoner map projections.

For quick reference, Hilliard *et al.* (1978) illustrate 22 projections, and with the aid of computer, show these projections in a variety of aspects. An indication of which special properties are preserved, if any, is included with each projection. A list of each projection's characteristics along with a statement of its intended use, method of transformation and historical origin is also provided. Similar information in the form of tables of the most important map projections is contained in Ray (1976), McDonnell (1979), Dent (1985), Cuff and Mattson (1982), Steers (1965), and Roblin (1969).

A number of articles exist on the application of map projections in certain fields and geographical areas. Bacon (1979) examines the criteria for selecting a projection for an atlas showing oceanographic data, and Gorton and Wainwright (1980) describe the basic properties and limitations of simple projections suitable for use in the presentation of similar information. This is particularly useful as an introduction as it assumes the reader has only a limited knowledge of projections and discusses the criteria for selecting suitable projections. Hoke *et al.* (1981) describe map projections and grid systems used for meteorological applications.

When reading texts or papers on projections, reference should be made to world and national atlases. For example, those published by Times Books (1985) and Oxford University Press (1973) provide clear illustrations of the use, characteristics and effectiveness of projections together with illustrated descriptions of the projections used. The American Congress on Surveying and Mapping has begun to publish a series of special publications giving guidance on the choice of projections (American Cartographic Association, 1986.).

Layout

Having decided upon the metrical framework for the map one must start to make decisions about the layout of the information to be included. This covers not only the geographic information, but also titles, legends, scale, and a wide range of marginal information that may be included. Generally there is considerable flexibility at this stage and, as Robinson *et al.* (1984) suggest, a series of rough sketches may be useful. The aim is to produce an overall balance in the final product by adjusting the position and contrast of the various components (e.g. Cuff and Mattson, 1982;

Dent, 1985). More formal layout tends to be required for series of maps or atlases, this being best described by Keates (1973).

Compilation

Compilation is the process of bringing together all the components required for the map into one document and it is here that the selection of methods for portraying the information must be finalized. The compilation process differs slightly for basic mapping, i.e. maps compiled from a survey carried out specifically for the production of a map, in comparison to compilation for derived mapping, i.e. maps where the locational information is derived from other, normally larger scale maps. Derived maps often also include information, particularly thematic data, from a variety of other sources.

Few cartographic textbooks concern themselves with basic mapping, Keates (1973, 1989) being the primary works in this area, although some others such as Anson (1988), Loxton (1980), Greenhood (1964), Render (1985) and Steele (1980) also give some coverage. Most of these discuss derived mapping as well. Some texts primarily devoted to surveying or photogrammetry discuss the compilation of basic maps, for example, Wright (1982) and Ritchie *et al.* (1988).

Most cartographic text books concentrate on the compilation and design of derived maps, mainly at smaller scales, and typically thematic, i.e. maps dealing with a specific topic or theme, rather than topographic or general purpose. The number of texts in this area has grown considerably in the last decade. Some are broad cartographic texts, such as Robinson, *et al.* (1984), Bertin (1981), Campbell (1984), Cuff and Mattson (1982), and Dent (1985). Older texts still of value include Robinson (1952), earlier editions of Robinson *et al.* (1969, 1978), Arnberger and Kretschner (1975), Cuenin (1973), Imhof (1972), Muehrcke (1972), Raisz (1962), and Witt (1970). Other texts are much more directed in their coverages such as Dickinson (1973), Hodgkiss (1970), Monkhouse and Wilkinson (1971) and Szego (1987). There is also a small number of books which look mainly at the nature and use of maps and also include some guidelines to their compilation (Muehrcke, 1978; Lawrence, 1979).

Depiction of map content

In creating a map we are in effect performing a transformation of some part of the earth's surface, and the phenomena pertaining to it, into an abstract representation using some form of symbols. This transformation from reality to map relies upon several processes. These are often set out in textbooks as apparently independent of one another, whereas in reality many of them are carried out simultaneously. The elements involved in this include the selection, classification and simplification of information to be mapped, and its graphic portrayal. The two main processes involved are usually referred to as symbolization and generalization. Opinions differ as to the relationship of these two processes, with one generally seen as subsuming the other, although how the processes are defined does not affect the outcome of design.

Any geographic phenomenon portrayed on the map must somehow be symbolized. How a particular feature is symbolized will depend not only on its nature, but also on the scale of the map. On all but the largest scale maps symbolization is also inextricably linked with generalization. It is not possible to map a feature with the full amount of detail that exists on the ground. Thus, the map symbol is a generalized representation of the feature, and/or some of its characteristics.

Symbolization

At this stage we will consider symbolization in its narrower sense, i.e. the use of the graphic variables for representation of features on a map. The design of symbols thus involves variations of form, dimension and colour, with colour itself having the three variables of hue, lightness and saturation. Some literature, particularly of North American origin, refers to the graphic variables as shape, size and colour, with colour having variations of hue, value and chroma.

It is not necessary to limit a symbol to one variable; frequently combinations are used. The variables employed will have great effect on our perception, and perhaps the most important aspect in designing symbols is the handling of contrast. Care must be taken to ensure there is sufficient contrast so that the user can perceive differences, but there must be some measure of balance, unless some specific element is to be emphasized.

In addition to the information they contain individually, it is important to recognize that map symbols also present information collectively; this can lead to much information being gained due to their distribution (Keates, 1973). Therefore symbols must not be designed in isolation, but the effects of their influence on the appearance of other symbols must also be considered.

Generally there are two means of classifying symbols: by their relationship to phenomena, or some aspect thereof, and by their use of graphic variables. In addition to the basic symbol types of are scaled according to some other value than their ground representation. Dent (1985) devotes more to this than most other texts and specific reference to the construction of cartograms may be found in Eastman, Nelson and Shields (1981), Hunter and Young (1968), Monmonier (1977) and Olson (1976).

Generalization

Generalization causes many problems in map creation. Basically what we are concerned with is the reduction in the amount of detail that can be shown as scale is reduced. As scale becomes smaller, so the area decreases, and therefore the number of features shown must also decrease if the map is to remain legible. This is not an arbitrary reduction by any fixed amount, as we must attempt to retain the general map content.

The main processes involved in generalization are selection, classification, combination and simplification. Exaggeration and displacement may also be required as scale is reduced. Selection and classification are normally largely carried out before the map is drawn, although some forms of selection will occur later. Combination, simplification and exaggeration are typically done during compilation. Most of the general cartographic texts devote considerable attention to generalization either in total or in treatment of its constitutent parts. Two works on generalization stand out. Steward (1974), while being largely theoretical is comprehensive in its coverage of the topic and has an extensive bibliography. *Cartographic Generalisation* (Swiss Cartographic Society, 1977) probably has the best collection of examples of treatments of the practical problems of generalization.

Selection is the process of deciding which information is to be included in the map (Robinson *et al.* 1984). This will depend primarily on the purpose of the map, but the number of categories we can include will decrease with scale. For example on a small scale map we are unlikely to be able to show both relief and vegetation. On a large scale map we may elect to show individual ancient monuments. Thus, there will be an elimination at smaller scales of categories which appear at larger scales. It is also likely that within individual categories some information may have to be removed. Although of primary importance in designing any map, selection has received little attention in the literature, some notable exceptions being Topfer and Pillewizer (1966), Unwin's (1981) discussion of drainage networks, and Kadmon (1972) and Stenhouse (1979) on the selection of settlements. Fitzsimons (1985) examines the selection of base information for thematic maps.

Classification refers to the grouping of data or their ordering or scaling (Robinson *et al.*, 1984). In the simple cases this may refer to the grouping of statistical data into a small number of discrete classes based upon some class interval scheme. It also refers to the classification of non-numerical information such as vegetation or soil types. The question of class intervals for statistical data has an extensive literature. It is well covered by texts such as Robinson, *et al.* (1984), Dent (1985), Dickinson (1973) and others. Articles of note include Coulson (1987), Evans (1977), Jenks and Caspal (1971), Jenks and Coulson (1963), Mackay (1955), Olson (1971), Stegena and Csilley (1987), and Tobler (1973).

Simplification is the determination of the important characteristics of a feature, the elimination of unwanted detail and the retention and possible exaggeration of the important aspects. Simplification refers mainly to lines – linear features and the boundaries of areas – but may also apply to points when symbolized by pictographic symbols. The simplification of lines has received much consideration in recent years due to the increased use of computers, with perhaps the best known algorithm being described by Douglas and Peucker (1973). The simplification of natural features such as coastlines and rivers attracts most attention, such as by Robinson *et al.* (1984), Marino (1979) and Pannekoek (1962).

Combination is a difficult element of generalization to quantify. A large scale map may show all areas of woodland. At smaller scale one could simply select the larger or more important areas, but a truer representation may result from the combination of several small areas into one larger area, often with a simplified boundary. The most comprehensive treatment is by the Swiss Cartographic Society (1977) and Keates (1973) deals with this to some extent. That combination is not appropriate for all classes of information is illustrated in Robinson and Sale (1969). The first of these three references also offers the best discussion of the problems of exaggeration and displacement.

Colour

The application of colour in maps is one of the most exciting yet least studied aspects of map design (Dent, 1985). However, its use in maps whether due to its impact on map legibility or aesthetic appeal, is often the source of criticism. Cartographers must therefore familiarize themselves with the basic principles of colour use, bearing in mind the purpose of the map.

Robinson (1952) suggests that the proper employment of colour relies on the cartographer being familiar with the physical and chemical laws relating to colour as light and pigment, the characteristics of human vision and perception, and the techniques of colour reproduction and printing. These elements are covered by some of the standard cartographic texts which provide sound introductions. Imhof (1982) provides a short review of the theory of colours and their classification. Robinson *et al.* (1984) describe the physical properties of additive and subtractive colour, the systems of colour specification and the principles of the selection of colours. Colour characteristics and specification systems together with human vision and perception are described by Keates (1973, 1989). Dent (1985) focuses throughout on the psychological and aesthetic aspects in his treatment of the use of colour and at the same time illustrates the reliance of cartography on other disciplines, chiefly physics, psychology and advertising. As Dent (1985) writes:

> 'Until detailed studies are provided, map designers need to follow the leads provided by merchandising, art, and related psychological literature.'

There are many works in the field of psychology to which a cartographer may wish to refer for guidance on how humans respond to colour both physiologically and psychologically. Birren (1978), Arnheim (1974), Evans (1974), Hilgard *et al.* (1975), Gregory (1977) and Sharpe (1974) write particularly informative and interesting introductions to the subject and offer general descriptions of such phenomena as colour interaction, colour preferences and the connotations of colour. These introductions tend to take colour in isolation and do not relate them to products such as maps or other marketable products. The advertising literature on the other hand may offer some relevant guidance. Standard introductory texts on advertising such as Engel (1980) and Burton and Ryan (1980) include sections on the use of colour.

Several cartographers have written short articles on the subject based on the type of research mentioned above. Sorrell (1974) comments on colour design for children's maps based on psychological research into colour preferences. Robinson (1967) provides a compilation of the more important of the perceptual aspects of colour and lists them in a logical structure. In so doing he provides a systematic summary of the psychological aspects of colour, a valuable introduction to those new to the subject.

An understanding of colour specification systems helps one to visualize the interrelationships of the three dimensions of hue, lightness (value) and saturation (chroma). The Munsell System (Birren, 1969a; Munsell, 1975) and the Ostwald System (Birren, 1969b) are being used increasingly by researchers in cartography in order to be more specific in their colour designations for papers and inks used in printing (Dent, 1985). The CIE System of colour notation may be of little direct benefit to practising cartographers as it is based on additive colour mixing and no colour charts are produced by the system. Nevertheless, cartographic researchers may find it useful in describing the physical attributes of colours. Kimerling (1980) believes that with new developments in map reproduction technology the transfer of map design guidelines from perceptual research, such as Cuff (1972) and Robinson (1967), to production cartography will be hindered until a widely accepted mathematical method for converting among systems is developed.

Relief portrayal

With the increased quality and quantity of elevation data, developments in reprographic techniques and the ever growing popularity of maps, the portrayal of the land surface form has taken on special significance in recent years. 'The proper rendering of terrain is one of the primary tasks of the cartographer' (Imhof, 1982).

Only relatively recently has literature become available that attempts to aid the cartographer solve the fundamental problem of recreating the three dimensional surface of the earth by two dimensional symbols. Terrain portrayal is often a source of criticism both for its failure to create an impression of relief in the mind of the map user but also for its interference with the legibility of other map information.

The authoritative work on cartographic relief representation is provided by Imhof (1982). Not restricted to pure descriptions of the various methods, this long awaited translation of his 1965 work provides an introduction to methods of topographical survey with special reference to accuracy in measuring terrain surfaces and the interpretation of relief from air photographs. Other concepts such as the theory of colours and the interplay of map elements form the foundation of the detailed treatment of the different methods of terrain representation. Accompanied by original map extracts, and beautifully illustrated throughout, this classic work is an example of design and production of the highest order.

In comparison other texts and papers may seem rather superficial. However, the standard cartography texts, notably Keates (1973), Robinson *et al.* (1984), Raisz (1962) and Monkhouse and Wilkinson (1971) provide adequate introductions. Brandes (1983) lists 48 relief representation techniques in an illustrated compendium designed to make cartographers more aware of rarely used techniques. References are given with each listed technique to enable the cartographer to select an appropriate technique for the depiction of specialized information. Ristow (1964) lists works that discuss and describe the application of three dimensional maps in civil and military contexts such as planning, recreation, flood control and mining. Toth (1973) gives an interesting insight into the 'human aspect' of the relief artist, together with some general principles of relief portrayal. Interesting articles on the use of hill-shading techniques are provided by Curran (1967) and Carmichael (1964). Both illustrate the advantages of using this technique in circumstances where speed and economy of production are essential. Although conventions in the use of colour for the portrayal of relief may never entirely disappear, Keates (1961) draws attention to the possibility of false subjective impressions of the environment being created by the conventional use of colour symbology. European cartography and the European physical background, Keates suggests, have played too great a part in the development of adopted conventions inappropriate for use in other parts of the world. Subsequently, relief portrayal has provided a subject for a number of experimentally based studies in map legibility and perception. Studies by Patton and Crawford (1977), Potash, Farrell and Jeffrey (1978), and Phillips, De Lucia and Skelton (1975) have sought to develop scientific design principles based upon the objective testing of cartographic products 'utilizing experimental techniques already established by human factors engineers, psychologists and physiologists.' (De Lucia, 1972). Their success is open to question. Nevertheless, reference to these works should at least alert the map designer to the possible effects that different relief portrayal methods may have on map information accessibility and interpretation, and more significantly to the diversity of the needs and abilities of the map user.

Lettering and name placement

> 'No matter how well designed a map might be in other respects, it fails if the lettering it depends upon for the transmission of its information is treated improperly' (Robinson, 1950).

Even though it is one of the more important aspects of cartographic design, lettering has received much less than its proper share of attention in the cartographic literature. Whether this is due to its familiarity, in books and newspapers, breeding contempt, or its lack of imaginative appeal, cartography still awaits an authoritative text on lettering. Dent (1985) writes:

> For the most part, the trained eye, and aesthetic judgement of the cartographer have produced the design strategies that are now standard.

The aims of a cartographer when selecting type styles and sizes have not changed since they were listed by Withycombe (1929). They are good legibility and harmony of style throughout the map and suitable selection of fount for the reproduction techniques employed. These aims form the basis for discussion in Hodgkiss (1966), Robinson *et al.* (1984) and Dent (1985).

The special problems of the use of type on maps have attracted very little attention amongst professional typographers. According to Keates (1958) much of the research in typography cannot be applied to lettering on maps. Nevertheless, proper selection of typeface and size can only be made if the cartographer understands the fundamentals of type design (Dent 1985). Typography textbooks such as Swann (1969), Gates (1976) and Lewis (1978) offer useful introductions to the elements and techniques of typography. The fundamentals of type are also covered by the standard cartography texts, notably Robinson *et al.* (1984), Keates (1973, 1989), Hodgkiss (1970), Raisz (1962), and Dent (1985).

Typeface guides such as Perfect and Routledge (1983) offer aids to typeface recognition and selection. In practice, however, type selection is normally limited to the typefaces available for the method of lettering production being employed and therefore reference may be limited to the appropriate catalogues or handbooks.

Perhaps the commonest kind of decision which must be made by the cartographer concerns the size of the lettering appropriate for the naming of the great variety of items on a map. Robinson (1950) addresses this problem bearing in mind the variations in visibility and legibility between type styles and the desire of the cartographer for type size to correspond with the hierarchical structure of the map. With the aid of tables, the selection of type sizes is discussed for original, wall and projected maps. Similar information is presented by Robinson *et al.* (1984) in the form of a nomograph. Shortridge (1979) makes several recommendations based on tests of people's ability to discriminate between type sizes in one type style. As Robinson *et al.* (1984) suggest, it is reasonable to assume that these recommendations can be extended to most other type faces. Brockemuehl and Wilson (1976) give very useful and specific directions for the minimum size of lettering for 35 mm slides and overhead projectors.

There is an extensive literature covering experiments on type legibility of books, newspapers and other printed matter. However, as Bartz (1970a) points out, the fundamental differences in the way type functions on a map make it difficult to generalize and apply conclusions drawn from these studies to map making. As a result, experiments have been conducted on type face as a factor contributing to name clarity on maps. Bartz (1970b), Phillips, Noyes and Audley (1977) and Foster and Kirkland (1971) are examples. Robinson (1952) is a useful reference for guidance on selecting typefaces and preserving lettering harmony. Suggestions are made as to the preferable ways of mixing type styles within the same fount and which different typefounts should be used if the need arises.

There is no doubt that a map can be greatly enhanced both aesthetically and functionally by good positioning of names. Imhof (1975) provides a series of precepts about positioning or locating the lettering on maps in relation to the various functional aspects of the map and the individual named features.

Imhof categorizes name placement into position, linear and areal designations and, with the aid of clear illustrations, provides a systematic guide to name placement. However, as he freely admits, there is no rule without exception and often rules may conflict. Recommendations based on experimental research by Noyes (1980) and Phillips, Noyes and Audley (1978), and Gerber (1982) largely support Imhof's suggestions and add further recommendations to cartographers in order to speed up word recognition time.

Conclusion

Our emphasis in this chapter has been to look at the practical design of maps, and whilst one cannot ignore the theory, most of the references cited do relate to the practice of making maps rather than research or hypothetical discussions.

There are two ways to view design: first as an innovative process creating new types of product, and secondly as a process of imitating existing designs. Most cartographic design falls into this latter category. Although the reader will find reference to the many writings discussed here of great benefit, no better advice can be given than to look at existing maps, see how the problems have been solved by others, and draw from their experience.

References

American Cartographic Association (1986). *Which map is best? Projections for world maps*. Falls Church, VA: American Congress on Surveying and Mapping

Anson, R. W. (1988). (Ed.) *Basic cartography for students and technicians, Vol. 2. London: Elsevier and ICA*

Arnberger, E. and Kretschmer, I. (1975). Wesen und Autgaben der Kartographie. Vienna: Deuticke

Arnberger, E. (1974). Problems of an international standardization of a means of communication through cartographic symbols. *International Yearbook of Cartography*. Vol. XIV pp. 19–33

Arnheim, R. (1974). *Art and visual perception*. Berkeley: University of California Press

Arnheim, R. (1976). The perception of maps. *American Cartographer* Vol. 3, No. 1. pp. 5–10

Bacon, C. J. (1979). Choosing the projection for an atlas. *Hydrographic Journal*. Vol. 16, pp. 21–28

Baker, A. (1986). Computers bring map projections back to life. *Geography*. Vol. 71(4) pp. 333–338

Balchin, W. G. (1954). The choice of map projections. *Empire Survey Review*. Vol. 12, pp. 263–276

Bartz, B. S. (1970a). An analysis of the typographic legibility literature: assessment of its applicability to cartography. *The Cartographic Journal*, Vol. 7, No. 1, pp. 10–16

Bartz, B. S. (1970b). Experimental use of the search task in an analysis of type legibility in cartography. *The Cartographic Journal*, Vol. 7, No. 2, pp. 103–112

Bertin, J. (1967). *Sémiologie graphique*. Paris: Gauthier-Villars

Bertin, J. (1981). *Graphics and graphic information processing*. Berlin: De Gruyter

Birren, F. (1969a). (Ed.) *A grammar of color: a basic treatise on the color system of Albert H. Munsell*, by T. M. Cleland. New York: Van Nostrand Reinhold

Birren, F. (1969b). (Ed.) *The color primer: a basic treatise on the color system of Wilhelm Ostwald*. New York: Van Nostrand Reinhold

Birren, F. (1978). *Color and human response*. New York: Van Nostrand Reinhold

Bollmann, J., Breithaupt, M., Dransch, W., and Freitag, U. (1988). *Thailand in maps*. Berliner Manuskripte Zur Kartographie. Berlin: Freie Universitat Berlin

Brandes, D. (1983). Sources for relief representation techniques. *The Cartographic Journal*, Vol. 20, No. 2, pp. 87–94

Brassel, K. E., Utano, J. J. (1979). Design strategies for continuous tone area mapping, *American Cartographer*, Vol. 6, pp. 39–50

Brockemuehl, H. W. and Wilson, P. B. (1976). Minimum letter size for visual aids. *Professional Geographer*, Vol. 28(2), pp. 185–189

Burton, P. W. and Ryan, W. (1980). *Advertising fundamentals*. Columbus OH: Grid Publishing

Campbell, J. (1984). *Introductory cartography*. Englewood Cliffs: Prentice-Hall Inc

Carmichael, L. D. (1964). Experiments in relief portrayal, *The Cartographic Journal*, Vol. 1, No. 1, pp. 11–17

Castner, H. W. and McGrath, G. (1971). (Eds.) *Map design and the map user Cartographica, Monograph* No. 2

Castner, H. W. and Robinson, A. H. (1969) *Dot area symbols in cartography: the influence of pattern on their perception*. A.C.S.M., Cartography Division Monograph No. CA-4. Washington: A.C.S.M.

Chang, K. (1980). Circle size judgement and map design, *American Cartographer*, Vol. 7, pp. 155–162

Coulson, M. R. C. (1987). In the matter of class intervals for choropleth maps. *Cartographica Monograph* 37. Vol. 24, No. 2, pp. 16–39

Cox, C. W. (1976). Anchor effects and thé estimation of graduated circles and squares, *American Cartographer*, Vol. 3, pp. 65-74

Cuenin, V. (1973). *Cartographie Générale*, Tome 1. Paris: Editions Eyrolles

Cuff, D. J. (1972). Value versus chroma in color schemes on quantitative maps, *Canadian Cartographer*, Vol. 9, pp. 134-140

Cuff, D. J. and Mattson, M. T. (1982). *Thematic maps: their design and production*. London: Methuen

Curran, J. P. (1967). Cartographic relief portrayal, *The Cartographer*, Vol. 4(1), pp. 28–38

Dahlberg, R. E. (1967). Towards the improvement of the dot map, *International Yearbook of Cartography*, Vol. VII, pp. 157-167

De Lucia, A. (1972). The effect of shaded relief on map information. *The Cartographic Journal*, Vol. 9, No. 1, pp. 14–18

Deetz, C. H. and Adams, O. S. (1945). *Elements of map projections*. United States Department of Commerce, Special Publication 68

Dent, B. D. (1985). *Principles of thematic map design*. Reading, MA: Addison-Wesley

Dickinson, G. C. (1973). *Statistical mapping and the presentation of statistics*, 2nd edn London: Edward Arnold

Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, *The Canadian Cartographer*, Vol. 10, pp. 112–122

Eastman, J. R. (1987). Graphic syntax and expert systems. *Technical Papers, A. C. S. M. Annual Convention*, pp. 87–96

Eastman, J. R., Nelson, W. and Shields, G. (1981). Production considerations in isodensity mapping, *Cartographica*, Vol. 18, pp. 24–30

Engel, J. (1980). *Advertising: the process and practise*. New York: McGraw-Hill

Evans, I. S. (1977). The selection of class intervals, *Transactions, Institute of British Geographers, New Series* 2:1, pp. 98–124

Evans, R. M. (1974). *The perception of color*. New York: Wiley

Fitzsimons, D. E. (1985). Base data on thematic maps. *American Cartographer*, Vol. 12, No. 1, pp. 57–61

Flannery, J. J. (1971). The relative effectiveness of some common graduated point symbols in the presentation of quantitative data. *The Canadian Cartographer*, Vol. 8, pp. 96–109

Forrest, D. and Castner, H. W. (1985). The design and perception of point symbols for tourist maps. *The Cartographic Journal*, Vol. 22/1, pp. 11–19

Foster, J. J. and Kirkland, W. (1971). Experimental studies of map typography. *Society of University Cartographers: Bulletin*, Vol. 6(1), pp. 40–45

Gates, D. (1976). *Lettering for reproduction*. Pitman.

Gerber, R. (1982). The young map user and the positioning of names on maps, *Cartography*, Vol. 12(3), pp. 162–167

Gorton, C. D. and Wainwright, P. (1980). Map projections in marine science, *Cartographica*, Vol. 17, No. 2, pp. 1–24

Greenhood, D. (1964). *Mapping*. Chicago: University of Chicago Press

Gregory, R. L. (1977). *Eye and brain: the psychology of seeing*, 3rd edn. London: Weidenfeld and Nicholson

Hilgard, E. R., Atkinson, R. C. and Atkinson, R. L. (1975). *Introduction to psychology*. New York: Harcourt Brace Jovanovich

Hilliard, J. A., Basoglu, U. and Muehrcke, P. C. (1978). *A projection handbook*. University of Wisconsin at Madison, Cartographic Laboratory Paper 2

Hodgkiss, A. G. (1966). Lettering maps for book illustration, *Cartographer*, Vol. 3(1), pp. 42–47

Hodgkiss, A. G. (1970 *Maps for books and theses*. Newton Abbot: David and Charles

Hoke, J. E., Hayes, J. L. and Renninger, L. G. (1981). *Map projections and the grid system for meteorological applications*. USAF Global Weather Center, Offutt AFB, NE, AFGWC/TN-79/003

Hsu, M-L. (1981). The role of projections in modern map design, *Cartographica*, Vol. 18, pp. 151–186

Hunter, J. M. and Young, J. C. (1968). A technique for the construction of quantitative cartograms by physical accretion models. *Professional Geographer*, Vol. 20, pp. 402–406

Hutchinson, M. F. (1982). MAPROJ – a computer map projection system. *Cartography*, Vol. 12(3), pp. 144–145

Imhof, E. (1975). Positioning names on maps. *American Cartographer*, Vol. 2, No. 2, pp. 129–144

Imhof, E. (1982). *Cartographic relief representation* (Translation of *Kartographische Geländedarstellung*, 1965). Berlin: Walter de Gruyter

Imhof, E. (1972). *Thematische Kartographie*. Berlin: Walter de Gruyter

Jenks, G. F. and Caspal, F. C. (1971). Error on choropleth maps: definition, measurement, reduction. *Annals, Association of American Geographers*, Vol. 61, pp. 217–244

Jenks, G. F. and Coulson, M. R. (1963). Class intervals for statistical maps. *International Yearbook of Cartography*, Vol. III, pp. 119–134

Jenks, G. F. and Knos, D. S. (1961). The use of shading patterns in graded series. *Annals of the Association of American Geographers*, Vol. 51:3, pp. 316–334

Joly, F. and De Brommer, S. (1966). Proposal for standardization of symbols on thematic maps. *International Yearbook of Cartography*, Vol. VI, pp. 47–79

Kadmon, N. (1972). Automated selection of settlements in map generalization, *The Cartographic Journal*, Vol. 9, pp. 93–98

Karssen, A. J. (1980). The artistic elements in map design, *The Cartographic Journal*, Vol. 17, No. 2, pp. 124–127

Keates, J. S. (1958). The use of type in cartography, *Surveying and Mapping*, Vol. 18, pp. 75–76

Keates, J. S. (1961). Techniques of relief representation, *Surveying and Mapping*, Vol. 21(3), pp. 459–463

Keates, J. S. (1972). Symbols and meanings on topographic maps. *International Yearbook of Cartography*, Vol. XII, pp. 168-181

Keates, J. S. (1973). *Cartographic design and production*. London: Longman

Keates, J. S. (1982). *Understanding maps*. London: Longman

Keates, J. S. (1984). The cartographic art, *Cartographica*, Vol. 21, No. 1, pp. 37–43

Keates, J. S. (1989). *Cartographic design and production*, 2nd edn London: Longman

Kellaway, G. P. (1970). *Map projections*. London: Methuen

Kilkoyne, J. (1974). Pictographic symbols in cartography. *Proceedings of the Association of American Geographers*, Vol. 6, pp. 87–90

Kimerling, A. J. (1975). A cartographic study of equal value gray scales for use with screened grey areas, *American Cartographer*, Vol. 2, pp. 119–127

Kimerling, A. J. (1980). Color specifications in cartography, *American Cartographer*, Vol. 7, pp. 139–153

Lawrence, G. R. P. (1979). *Cartographic methods*, 2nd edn. London: Methuen

Lewis, J. (1978). *Typography: design and practise*. New York: Taplinger

Loxton, J. (1980). *Practical map production*. Chichester: Wiley

Mackay, J. R. (1955). An analysis of isopleth and choropleth class intervals, *Economic Geography*, Vol. XXXI, pp. 71–81

Mackay, R. R. (1949). Dotting the dot map: an analysis of dot size, number, and visual tone density. *Surveying and Mapping*, Vol. 9, pp. 3–10

Maling, D. H. (1968). The terminology of map projections. *International Yearbook of Cartography*, Vol. VIII, pp. 11–23

Maling, D. H. (1973). *Co-ordinate systems and map projections*. London: George Philip

Maling, D. H. (1984). Mathematical cartography. In *Cartography for students and technicians*, edited by Koeman, C. *et al.* Vol. 1. International Cartographic Association, pp. 32-78

Marino, J. S. (1979). Identification of characteristic points along naturally occurring lines, *The Canadian Cartographer*, Vol. 16(1), pp. 70–80

Marschner, J. L. (1944). Structural properties of medium and small scale maps. *Annals of the Association of American Geographers*, Vol. 34, pp. 1–46

McDonnel, P. W. (1979). *Introduction to map projections*. New York: Marcel Dekker

Meihoffer, H. J. (1969). The utility of the circle as an effective cartographic symbol. *The Canadian Cartographer*, Vol. 6, pp. 105–117

Modely, R. (1970). Universal symbols and cartography. In *Symposium for Map Use and the Map Reader*, edited by Castner, H. W. and McGrath, G. Queen's University Canada

Modely, R. (1976). *Handbook of pictorial symbols*. New York: Dover Publications

Monkhouse, F. J. and Wilkinson, H. R. (1971). *Maps and diagrams*. London: Methuen

Monmonier, M. S. (1977). *Maps, distortion, and meaning*. Resource Paper No. 75–4. Washington: Association of American Geographers

Morrison, J. L. (1974). A theoretical framework for cartographic generalization with emphasis on the process of symbolization, *International Yearbook of Cartography*, Vol. XIV, pp. 115–127

Muehrcke, P. (1972). *Thematic cartography*. Resource Paper No. 19, Association of American Geographers 1972

Muehrcke, P. C. (1986). *Map use*, 2nd edn Wisconsin: JP Publications

Muller, J.-C., Johnson, R. D. and Vanzella, L. R. (1988). A knowledge-based approach for developing cartographic expertise. *Proceedings 2nd International Symposium on Spatial Data Handling*, pp. 557–571

Munsell, A. H. (1975). *A color notation*, 12th edn Baltimore: Munsell Color

Newton, G. D. (1985). *Computer programs for common map projections*. USGS Bulletin 1642

Noyes, E. (1980). The positioning of type on maps: the effect of surrounding material on word recognition time. *Human Factors* Vol. 22(3), pp. 353–360

Olson, J. (1971). The effects of class interval systems on choropleth map correlation. *Proceedings, Association of American Geographers*, pp. 127–130

Olson, J. (1976). Noncontiguous area cartograms. *Professional Geographer*, Vol. 28, pp. 371–380

Olson, J. (1977). Rescaling dot maps for pattern enhancement. *International Yearbook of Cartography*. Vol. XVII, pp. 125-137

Oxford University Press, (1973). *The Oxford world atlas*. London: Oxford University Press

Pannekoek, A. J. (1962). Generalization of coastlines and contours. *International Yearbook of Cartography*, Vol. II, pp. 55–74

Patton, J. C. and Crawford, P. V. (1977). The perception of hypsometric colours. *The Cartographic Journal*, Vol. 14, No. 2, pp. 115–127

Perfect, C. and Routledge, G. (1983). *Routledge's international type finder: the essential handbook of typeface recognition and selection*. London: Sarema Press

Phillips, R. J., De Lucia, A. and Skelton, N. (1975). Some objective tests of the legibility of relief maps, *The Cartographic Journal*, Vol. 12, No. 1, pp. 39–46

Phillips, R. J., Noyes, L. and Audley, R. J. (1977). The legibility of type on maps, *Ergonomics*, Vol. 20, pp. 671-682

Phillips, R. J., Noyes, L. and Audley, R. J. (1978). Searching for names on maps, *The Cartographic Journal*, Vol. 15, No. 2, pp. 72–76

Potash, L. M., Farrell, J. P. and Jeffrey, T. (1978). A technique for assessing map relief legibility, *The Cartographic Journal*, Vol. 15, No. 1, pp. 28–37

Pustowski, R. (1977). The standardization of symbols for tourist maps and the problems in relevant co-editing publications. Paper presented to *ICA Commission IV*, Hamburg

Raisz, E. (1962). *Principles of cartography*. New York: McGraw-Hill

Ratajski, L. (1971). The methodological basis of standardization of signs on economic maps, *International Yearbook of Cartography*, Vol. XI pp. 137–159

Ray, J. M. (1976). Properties of map projections. *Special Libraries Association, Geography and Map Division Bulletin*, 105, pp. 6–11

Render, J. (1985). *Monochrome map preparation*. Society of University Cartographers

Ristow, W. W. (1986). *Three dimensional maps: an annotated list of references related to the construction and use of terrain models*. Washington DC: Library of Congress, Map Division, Reference Department. 1964

Ritchie, W., Tait, D. A., Wood, M. and Wright, R. (1988). *Mapping for field scientists*. Newton Abbot: David and Charles

Robinson, A. H. (1950). The size of lettering for maps and charts. *Surveying and Mapping*, Vol. 10, pp. 37–44

Robinson, A. H. (1952). *The look of maps*. Madison: The University of Madison Press

Robinson, A. H. (1967). Psychological aspects of color in cartography. *International Yearbook of Cartography*, Vol. VII, pp. 50–61

Robinson, A. H. (1970). Scaling nonnumerical map symbols. *Proceedings, A. C. S. M. Annual Convention*

Robinson, A. H. (1973). An international standard symbolism for thematic maps: approaches and problems, *International Yearbook of Cartography*, Vol. XIII, pp. 19–25

Robinson, A. H. (1974). A new map projection: Its development and characteristics. *International Yearbook of Cartography*, Vol. XIV, pp. 145–155

Robinson, A. H. and Sale, R. D. (1969). *Elements of cartography*, 3rd edn. New York: Wiley

Robinson, A. H., Sale, R. D. and Morrison, J. L. (1978). *Elements of cartography*, 4th edn New York: Wiley

Robinson, A. H. *et al.* (1984). *Elements of cartography*, 5th edn. New York: Wiley

Roblin, B. A. (1969). *Map projections*. London: Edward Arnold

Rouleau, B. (1984). Theory of cartographic expression and design. *Basic Cartography for Students and Technicians*, Vol. 1. ICA

Sharpe, D. (1974). *The psychology of color and design*. Chicago: Nelson-Hall

Shortridge, B. G. (1979). Map reader discrimination of lettering size. *American Cartographer*, Vol. 6, No. 1, pp. 13–20

Snyder, J. P. (1984). *Computer-assisted map projection research*. USGS Bulletin 1962

Snyder, J. P. (1986). *Map projections used for the large-scale quadrangles by the USGS*. USGS Circular 982

Snyder, J. P. (1987). *Map projections – a working manual*. USGS Professional Paper 1395

Sorrel, P. (1974). Map design – with the young in mind, *The Cartographic Journal*, Vol. 11, pp. 82–91

Steele, R. C. (1980). *Modern topographic drawing*. Houston: Gulf
Steers, J. A. (1965). *An introduction to the study of map projections*. London: University of London Press
Stegena, L. and Csillag, F. (1987). Statistical determination of class intervals for maps, *The Cartographic Journal*, Vol. 24, pp. 142–146
Stenhouse, H. (1979). Selection of towns on derived maps, *The Cartographic Journal*, Vol. 16, No. 1, pp. 30–39
Steward, H. J. (1974). Cartographic generalization: some concepts and explanation. *Cartographica* Monograph No. 10
Swann, C. (1969). *Techniques of typography*. London: Lund Humphries
Swiss Cartographic Society (1977). *Cartographic generalisation*. Zurich: Swiss Cartographic Society
Szego, J. (1987). *Human cartography: mapping the world of man*. Swedish Council for Building Research
Taylor, D. R. F. (1983). (Ed.) *Graphic Communication and Design in Contemporary Cartography*. Chichester: Wiley
Times Books (1985). *Times Atlas of the World*, 7th Comprehensive Edition. London: Times Books
Tobler, W. (1973). Choropleth maps without class intervals. *Geographical Analysis*, Vol. 5, pp. 262–265
Topfer, F. and Pillewizer, W. (1966). The principles of selection, a means of cartographic generalization. *The Cartographic Journal*, Vol. 3, pp. 10–16
Toth, T. G. (1973). Terrain representation past and present at the national geographical society. *Proceedings A. C. S. M. Fall Technical Meeting*, pp. 9–31
Unwin, D. (1981). *Introductory spatial analysis*. London: Methuen (UP)
Williams, R. L. (1958). Map symbols: equally appearing intervals for printed screens. *Annals*, Association of American Geographers, Vol. 48, pp. 63–84
Wilson, E. (1967). The design and selection of graded shadings for black and white maps, *Bulletin of the Society of University Cartographers*, Vol. 1, pp. 11–16
Withycombe, J. G. (1929). Lettering on maps. *Geographical Journal*, Vol. 73(5), pp. 429–446
Witt, W. (1970). *Thematische Kartographie*. Hanover: Gebrüder Janecke
Wright, J. (1982). *Ground and air survey for field scientists*. Oxford: Clarendon Press

CHAPTER TWELVE

Contemporary map production

JOHN ROBERTSON

Map production and reproduction

The fact that a map needs to be produced and then reproduced, in some form, to give multiple copies, has always been a design constraint for cartographers. Maps have two features that tend to distinguish them from other allied graphical products: firstly, they tend to be drawn colour separate and not amalgamated until the final prints; and secondly, they need to be planned backwards with both the reprographic and production means in mind. There is some difficulty with the terms themselves as a suitable all embracing word does not seem to exist. It is generally accepted that map production covers the operations up to the 'camera ready' copy stage, i.e. draughting, scribing, masking et cetera. These procedures are usually well treated in the literature and text books. Reproduction refers to the processing of multiple copies and includes photomechanical and printing processes. This is less well documented in cartographic terms though there is a host of specialized knowledge within the printing trade that can be applied. Some excellent recent publications (Brannon, 1988; Curran, 1988; Doverlodge, 1988 and International Cartographic Association, 1988) have tended to redress the balance.

Map production articles and sections within books often tend to deal with the means and the methods very well. The actual evaluation of materials and equipment is less prevalent. Partly this is because specific items would be superseded, affecting the currency of any publication. Also many of these assessments are very subjective as they depend on individual tastes and preferences. In this connection 'self trial' is often the only real method of

A MODEL OF CARTOGRAPHIC DESIGN FOR EXPERT SYSTEM APPLICATIONS.

David Forrest
Department of Geography
Portsmouth Polytechnic
PORTSMOUTH PO1 3HE, U.K.

ABSTRACT

The continual drop in price and enhancement in performance of computer hardware has brought the possibility of computer mapping to a wider audience. This increase in the availability of computer mapping capabilities has lead to a rise in the number of potential map authors, but does not however appear to have lead to more widespread knowledge of cartographic design theory. As it is unlikely that the general level of cartographic education of most computer map authors will be greatly increased, cartographers must strive to make the programs used by naive map authors better able to produce reasonably well designed maps, or at least maps which do not break the most fundamental rules of map design. To enable such a system to be developed the basic map design process must be formalised. This is seen as a preamble to developing specific rules which will be incorporated into a prototype cartographic design expert system.

INTRODUCTION

In the last twenty years or so there have been a great number of programs created to produce maps using a computer.[1] Most of the commonly available programs are for producing small scale statistical maps, however more recently there has been a significantly increased interest in Geographic Information Systems for producing a wider variety of maps at a broader range of scales. The continual drop in hardware prices has brought the possibility of computer mapping to a wider audience, particularly with the increased power of micro-computers now available.

The increase in the availability of computer mapping capabilities has lead to a great increase in map authors[2] and the number of maps being produced, but does not however appear to have lead to more widespread knowledge of cartographic design theory. The large number of poorly designed maps created by map authors using computer systems to produce their own maps indicates that there is a lack of knowledge of how to design maps. These poorly designed maps are not the fault of the computer programs. Most programs do have the capability of producing well designed maps when used by someone knowledgeable in map design. The problem lies with authors who are not skilled in cartographic design and who would probably never produce a map by conventional means, but would contract a cartographer to produce it.

It is unlikely that the general level of cartographic education of most computer map authors can be greatly increased, nor than computer map design can be limited to those with cartographic training, therefore cartographers must strive to make the programs used by non cartographers better able to produce reasonably well designed maps, or at least maps which do not break the most fundamental rules of map design. The area of computer science devoted to producing programs that include knowledge of how an expert solves a problem is that of Expert Systems. An Expert System is essentially a program which includes a codified form of the rules that an expert uses to solve a problem, thus a cartographic design expert system would include the rules a cartographer uses when designing a map.

A long term goal would be to have a cartographic design expert system that could design any map at any scale, but current literature on expert systems suggests that at this time practical expert systems should be limited to narrow domains, ie. the problem area must be well defined within quite narrow margins. Several cartographic expert systems are currently under development. These have tended to concentrate on elements of the map or map design, eg name placement, line generalisation, solution of spatial conflicts, and while all these problems will have to be solved in any realistic production system, there would seem to be a need for the application of expert system techniques to more general design issues such as who the intended user is, why the map is required, what its intended use is, where it is to show, what information is required, how to represent the information in map context, and generally trying to prevent the author from making poor design decisions when making the map.

Several papers on cartographic design expert systems seem to assume that the process starts with the symbolisation of the map information. Clearly an examination of the map design process shows that this comes relatively late in total process. The system described here commences with questions about the nature of the map and its intended purpose, establishing a background to the map. Discussions with cartographers have indicated that this could go further, or perhaps even back a step to question the map author as to why he wants a map in the first place and what he expects it to achieve. While this approach may be desirable, until we have much more sophisticated natural language interpreters it is unlikely that this will be achievable in any realistic sense with a flexible form of dialogue.

In the following discussion some very broad generalities are stated. These are not intended to be specific solutions to all situations that may occur in cartographic design, but descriptions of some of the problems that may face the prototype system design, and their possible solution. This is seen as a preamble to developing specific rules. It is recognised that the formation of rules in some cases is trivial, but that others may require considerable effort and experimentation.

[1] Numerous terms have been used to describe maps produced with the aid of computers: computer cartography, computer mapping, computer assisted cartography, digital mapping, automation, etc. In the current study the term Computer Aided Cartography (cf. Computer Aided Design) will be used to refer to computers being used for the design and display of maps, whereas Computer Mapping will be used to refer to the broader use of computers in map making.

[2] The **Map Author** is one who conceives the map and prepares the data. He may then proceed to carry out the design and production, or pass this on to the **Cartographer**. The **Map User** is the intended audience of the map. This may or may not include the Map Author. The **System User** is the user of a Computer Mapping system. He may be the Map Author, the Cartographer, or the Map User.

CHOICE OF SUBJECT

To create a narrow domain cartographic design expert system one must select the type of maps and an appropriate scale range to be examined. With the forthcoming availability of large spatial data sets such as the World Digital Database for Environmental Science, it would seem appropriate to focus on the application of this type of data, thus this paper parallels the traditional map design procedures used for designing maps in small scale national or regional atlases with a model of cartographic design for an expert system for producing a similar range of maps. Possible products would include maps of the physical or cultural landscape at scales from 1:2 million to 1:15 million. Comparable conventional printed maps and atlases already exist, e.g. the home country and regional maps in products like "A Senior Secondary Atlas for Nigeria" by Collins-Longman (1983). In addition to small scale topographic maps, these atlases have a wide range of special topic maps, such as relief maps, land use, communication, climate, etc, which include point, line and area information in both numerical and nonnummerical form, thus rules will be required to deal with most types of information found on maps.

In order to create a cartographic design system, the first step must be to formalise the map design procedure so that it can be translated into an appropriate knowledge representation scheme. Unfortunately there is little in the cartographic literature relating to such a formalism for the whole process, design largely being the result of experience or imitation. Thus rather than following a single source, the model is based on practical experience combined with theoretical elements from Bertin (1967), Keates (1989), Robinson et al. (1984) and others (see Forrest and Pearson, in press).

As the map is a representation of some of the earth's phenomena, a fundamental part of the model must examine the relationship between the various types of phenomena that can exist, the data available about them and their possible cartographic representations. Although a database containing base data and special topic data would be part be part of a final system, allowing a variety of maps to be produced, the map author must also be able to add special topic information for the system to be of more general use. Thus in addition requiring formalised knowledge about the information in the internal database, the system must be capable of interacting with the author to allow him to describe the nature of other the phenomena to be mapped and the information available about it with a view to assigning appropriate cartographic representations to each data set. This could be achieved by a self contained expert system of the diagnostic/classification type, the most common type of expert system, and probably the easiest to develop. Space does not allow further elaboration of this aspect here, but it is seen as a fundamental part of the formalisation process.

THE DESIGN PROCESS

The general procedure for designing a map follows the route of: Compose; Compile; Symbol specification; Image construction; Adjust. A similar route can be followed with computer aided cartography, and indeed it is logical that an expert system follow a similar route to a cartographic expert. In some aspects there is a direct parallel between traditional and computer aided map design, but there are necessarily some differences due to the change in media. The stages to be followed

here are: Description; Layout; Data Selection; Symbolisation; Display; Modify. These are described in more detail below.

Description

In this step the aim is for the user to describe to the system some basic information about the map he requires.

First then, the user must inform the system of the type of map to be produced. Immediately a distinction can be made between topographic or 'general purpose' maps and special topic maps. A topographic map ideally shows all information with the same level of importance, ie no one aspect of the map should dominate (although this is not always the case in practice) whereas for a special topic map, the thematic information normally will be the dominant part of the graphic image. Thus, the system will have to know what type of map is required at an early stage.

The system will have 'knowledge' about a number of map types that can be produced from the information in its database, however the user may not be familiar with such titles and may require help in defining what he wants. He may indeed require a map that the system does 'know' about, but refer to it by a different title. From the author's description of the information to be included in the map the system should be able to determine the basic type of map required. The user's name for this may be added to the knowledge base for future reference.

For some maps the system will be able to exercise almost total control over the map design as all the information will be contained in the system data base and the system 'knows' about this type of map. If an 'unknown' map type is required, more information will be needed from the user, particularly if the main information to be mapped is not included in the system database. In this event the user will have to describe the phenomena to be mapped, the data available, and supply the necessary spatial and/or non spatial data.

The purpose for which the map is to be used is also important in determining what information should be included in the map, the level of detail which may be required and hence the scale that will be required to show the desired detail. As the proposed system is specifically limited to producing small scale maps in a particular scale range, there are obviously limits on the intended use of the output. It is sufficient here to differentiate between maps required as an overview and those required for more detailed inspection.

In addition to this general statement of intended use, the map author should be given the opportunity to specifically state the level of detail required. This can be allocated a number in the range 1-5 indicating an increasing level of detail. The system will check for conflicts intended use, level of detail and scale.

The intended user of the output should also be considered at this stage. The map author can also be the map user, but the map may be being produced for a wider audience. If the author is also the intended user then one can normally assume some familiarity with the location or information being portrayed. If the map is intended for a wider audience they may or may not be knowledgeable about the area or subject. If the map is intended for a naive audience it may be desirable to make the map simpler than for an expert audience. This will reflect on both the content of the map and the level of detail at which each element is shown.

Layout

The factors to be decided at this stage are the actual geographical area to be mapped; the format of the output; and the scale of the output. A decision on the first of these and one other will determine the third. The level of detail determined above may influence the choice of scale. Some backtracking may be required if it is not possible to show the desired area in the available format at a scale commensurate with the topic or the level of detail requested.

Location. As a general point, the map author will know within reasonable limits the area to be mapped, although the size and shape of the area of primary interest and the purpose of the map may influence how much of the surrounding area will be included. It should be possible to specify the area by name (country or region), latitude and longitude, projection coordinates, or visually from a map display. It is unlikely that the system will be of much assistance in determining the desired area, although help can be given in the form of an interim plot of the area of interest, allowing a more precise definition of the boundaries.

Format. If the map is only to be displayed on a CRT display then the format will most likely be the maximum size allowed on the display. If hard copy is required this will be limited by size of printer or plotter to be used, but may also be limited by other factors, such as the page size of a publication. If the map is for reproduction the author may have fixed criteria. Help at this stage would be in the form of basic rules of appearance, such as ratio of sides, etc, and selection of portrait or landscape format depending on the shape of the area of interest.

Scale. The general principle with atlas maps is to use the maximum scale possible within the given format, thus this will normally be calculated by the system based upon the size if the area and the format. It is possible that the user will require the map to be of a certain scale (if one of a series perhaps), thus once scale and location have been specified the required format can be calculated.

Scale is a topic that is frequently misunderstood by people, particularly at the small scales used here so it is likely that the system will have to provide a considerable amount of explanation. The interaction between location, format and scale often leads to difficulties. Frequently authors want a scale larger than the format will allow, so some compromise will be required. The system may have to backtrack and inform the user of any changes resulting from such conflicts.

Marginal Information. The proposed system will not at this time make any attempt to design the layout of marginal information such as titles, legends, etc., but will use a standard layout. Reserving space for these items can influence the scale or format of the map, but the infinite number of possible arrangements is largely beyond the scope of this project. It is conceivable that future systems could analyze the 'free space' around the map and offer some assistance with layout.

Data Selection

The information to be included in a given map will depend upon the type of map, the scale, and the level of detail required. Table 1 illustrates a sample of the information to be included in the system database. Each data set is ranked for inclusion in each known map type. If a known map type was specified in the Description module the system will be able to select the appropriate data sets automatically. Alternatively, lists of the available information can be displayed for the user to select.

Special Subject Maps. If a special subject map is being produced, more information will be needed from the user, particularly if the main information to be mapped is not included in the system data base. The user will be prompted to describe the phenomenon and the data he has available at this stage. This can in fact be seen as a separate task to the design aspects of the system and fits the model of a classification expert system as noted above.

Base Information for Special Subject Maps. In designing a special subject map one must have an appropriate base map on which to display the information. There are two common approaches to this. The first is to take a topographic map and reduce it to a background image, often by printing it in grey. This will mean that much superfluous information will be included and also that some essential information may be obscure. The second approach is to design the base specifically for the map. This involves selecting the appropriate information from the topographic base and symbolising it to compliment the special topic information. This should result in a better solution, and is the obvious approach to take with a digital database.

Frequently little consideration appears to be given to the level of detail of the base image when compared to the special topic representation. For example, a map with a very detailed coastline showing very generalised climatic information can mislead the user into thinking the special topic information is as detailed as the topographic information. Thus, some attempt must be made to have commensurate levels of detail for different elements of the map. This may involve simplifying the base information to more closely reflect the detail or accuracy of the special topic information.

Map Complexity It is very easy for a map to become very complex. Generalisation is the process used to reduce this complexity, and posses many problems in map design. The selection aspect of generalisation has been incorporated, but the need to simplify information, particularly lines can be avoided for the indicated scale ranges by holding linear data at two generalisation levels and allowing the system to select the most appropriate, given scale and level of detail. Problems of overlapping point representations may arise at smaller scales, but rules such as those developed by Mackaness (1986) could be incorporated.

Symbolisation

Having made an initial selection of the information to be included in the map (which may have to be modified once the map has been displayed) each element of the map will have to be given a symbol specification. The actual details of this will vary quite considerably depending on the phenomena being mapped, the scale, etc., however the first step is to assign the cartographic representation to be used. Each of the data sets to be included in the data base may be assigned one or more possible representation based upon the nature of the phenomena, the locational data and its level of measurement as discussed above. The possible cartographic representations are indicated in Table 2.

This is only the first step in specifying the symbols. Once the type of representation is known, specific symbols will have to be assigned to the information. In some cases this will be trivial, such as specifying colour and gauge of rivers. In other cases considerable effort will be required to select the most appropriate set of point symbols, or colour scheme for example. Rules for the representation and for the data set will be used to narrow the choice, but inevitably user choice will play a major role here. For example, for a temperature map the system knows this is a bi-polar numerical distribution and would select a scale of blues for the cooler temperatures and a reds for the warmer temperatures. It may, however, be advisable for the user to indicate the change over point rather than this simply being the middle of the scale.

Feature coded information, particularly area coverages, present one of the biggest difficulties here, as often there are a large number of categories. The available colour space could be divided up to maximise the differences, although the results may be unpleasing. A more productive approach would be to let the user select from a palette or series of palettes, with the system ensuring that noticeable differences are maintained (Samson and Poiker, 1985).

While perhaps to be avoided for reasons of simplicity, it is possible that a map author will require a map showing more than one topic. (The trend of bivariate mapping of statistical / census information is not included in this discussion.) The map may for example show both temperature and precipitation. As there are strict limits on the number of continuous phenomena that can be shown by area symbolism - probably two at most, one being shown by area colours and the other by area patterns - the user will have to prioritise the phenomena to be depicted, and may have to opt for line or point symbols to depict some continuous phenomena, eg the use of isolines. For example, a map could show annual precipitation by layer tints and January and July isotherms by two sets of lines. This type of checking for conflicts between representations is an integral part of the system.

Display

Having determined the information to be included and its graphical representation the map can be displayed on the screen.

Modify

It would be ambitious to suppose that the first attempt will be exactly what the user requires, thus the system must be able to interact with the user to enable modifications to the design. The biggest difficulty here is in assessing what good design is, as this is largely subjective and attempts at quantifying this (e.g. Mackaness et al. 1986) bear little resemblance to the user's perceptual response.

CONCLUSION

The main aim of the cartographic design expert system discussed here is to prevent fundamental rules of map design being broken. No attempt is made to develop a creative design tool, rather the emphasis is on imitating the design of existing maps that are effective in solving the problems associated with mapping various types of information. The formalisation of the rules of cartographic design is the critical step. Once these have been finalised they will be incorporated into prototype system written in Turbo PROLOG.

Table 1 Proposed Database Contents.

BASIC INFORMATION

Political/Administrative Boundaries - International and internal (2 levels if available). These will be used with separate census data file for statistical maps.

Coastline - similar level in hierarchy as International boundaries. Should be able to be shown at two levels of generalisation.

Drainage - network classified with at least 3 levels.

Lakes - large lakes (say greater than 2mm^2 at the largest map scale). Must be linked with drainage network.

Railways - one level.

Roads - classified as highways/motorways, major roads, other roads.

Settlements - (administrative definition) database would contain classification based on simple hierarchy, e.g. National capital, State capitals, other cities and important towns, giving 4 levels of hierarchy.
Ideally settlements would be chosen from a separate database with a variety of factors, e.g. population, political status, remoteness, etc., with a ranking calculated from these. Cutoff point determined by scale initially. User could specify number of settlements to be included, or selection criteria based on facts in database and system make choice. Different default parameters could be specified for different map types. Number of categories dependent on scale, number of settlements to be included and map type.

Contours - frequently shown on atlas maps with non uniform interval, as basis of layer colours. Database should include all contours based upon minimum interval appropriate for region. Actual intervals used selected automatically depending upon scale. Area symbolisation may be included depending on the type of map.

SUPPLEMENTARY INFORMATION

(Information frequently used for special topic maps in regional atlases)

Geology, Soils, Land Use, Land Cover (Vegetation), Annual Precipitation, Average Temperature.
Census data, including population etc. (to be used with administrative boundaries above)

Table 2 **CARTOGRAPHIC REPRESENTATIONS**

POINT SYMBOLS

0A Dot Distribution. All points have same value.
a represents occurrences of discrete individuals
b represents some fixed quantity grouped for each dot

0B Classified. All features of same importance. Data feature coded. Geometric or pictographic symbols

0C Univariate Quantitative. Symbols scaled according to some value. May represent points, areas considered as points at map scale or areas represented at point location (usually zone centroid)
a unipolar distribution
b bipolar distribution

0D Multivariate Quantitative. Symbol scaled to some value or fixed. May represent points, areas considered as points at map scale, or areas (usually by zone centroid).

0E Digital Surface Model. Series of point locations with values. May have regular or irregular distribution.

LINE SYMBOLS

1A Boundaries. Can represent:
natural boundaries (often zone of change)
actual ground feature (often man made)
intangible (no real existence on the ground)
a nominal - all of same value/importance - may be feature coded.
b hierarchical - typical of political/administrative boundaries.

1B Networks.
a feature coded, often hierarchical networks e.g. roads.
b branching hierarchical networks e.g. river systems.

1C Isolines. (contours) Divides surface into zones between two values.

1D Flow lines. Can be in form of network, or independent routes/movement

AREA SYMBOLS - each zone homogeneous; changes occur at zone boundaries.

2A Isolated areas. Binary division of surface - special case of 2C.

2B Unclassed. ('Colour patch map', 'Chorochromatic') e.g. 'Politcal' map.
a uniform
b hierarchical e.g. Country/state/county

2C Classified. (Categorical, area-class) Data feature coded. e.g. Land use, vegetation, geology.
a uniform
b hierarchical (i.e. sub-classes)

2D Graded. Two possibilities of boundaries, but treatment symbolisation the same:
Choropleth Discrete zones with numerical values - boundaries usually imposed on distribution (normally administrative)
Dasymetric Discrete zones with numerical values - boundaries derived from distribution.
a unipolar distribution
b bipolar distribution
c bivariate (not Dasymetric)

2E Layered. (e.g. hypsometric colours)
a unipolar distribution
b bipolar distribution (eg temperature zones)

2F Cartograms. Areas scaled by a variable, not true extent.

SURFACE SYMBOLS - differ from areas in that continuous variation is represented.

3A Shading. (e.g. hill shading)

3B Block Diagrams. (Arguably not a true map - non orthogonal)
a continuous
b stepped

3C 3D Surface Model. (cannot be displayed in this form - really an internal representation in a computer or a physical model, however the visualisation of the 3D surface could be considered in this class)

References

Bertin, J. (1967) Semiology Graphique. Paris: Gauthier-Villars.
Forrest, D., Pearson, A.W. (In Press) "Information Sources in Map Design" In Parry and Perkins (eds.) Information Sources in Cartography. London: Butterworth.
Keates, J.S. (1989) Cartographic Design and Production (2nd Edition). London: Longman.
Mackaness, W. (1986) "Detection and Heuristic Resolution of Spatial Conflicts in Digital Map Datasets." Unpublished report, Kingston Polytechnic.
Mackaness, W., Fisher, P., Wilkinson, G. (1986) "Towards a Cartographic Expert System." Proc. Auto Carto London Vol. 1, pp.578-587.
Samson, L, Poiker, T.K. (1985) "Graphic Design with Color using a Knowledge Base." Paper presented at the 10th C.C.A. Annual Conference.
Robinson, A., Sale, R., Morrison, J., Muehrcke, P. (1984) Elements of Cartography (5th Edition). New York: Wiley.

EXPERT SYSTEMS FOR CARTOGRAPHIC DESIGN AND PRODUCTION

David Forrest

If the number of publications using the terms in their title is any indication, then in the last few years there has been considerable interest shown in the application of artificial intelligence and expert systems to cartography. Most are theoretical considerations of what can and might be done, or what can't, and while a few do report actual working examples, these are all of fairly limited scope or sophistication. A small number of publications try to deal with the general problem of map design, or significant portions of it, but more are concerned with specific aspects of map design and production, such as line simplification, name placement, or symbol selection. This paper introduces the basic concepts of artificial intelligence and expert systems and examines what needs to be done to apply them to cartography.

INTRODUCTION

In the last twenty years or so there have been a great number of programs created to produce maps using a computer[1]. Most of the commonly available programs are for producing small scale statistical maps, however more recently there has been a significantly increased interest in Geographic Information Systems for producing a wider variety of maps at a broader range of scales[2]. The continual drop in hardware prices has brought the possibility of computer mapping to a wider audience, particularly with the increased power of micro-computers now available.

The increase in the availability of computer mapping capabilities has lead to a great increase in map authors[3] and the number of maps being produced, but does not however appear to have lead to more widespread knowledge of cartographic design theory. The large number of poorly designed maps one sees that have been produced by computer mapping systems indicates that there is a general lack of knowledge of how to design maps. These poorly designed maps are not the fault of the computer programs. Most programs do have the capability of producing well designed maps when used by someone knowledgeable in map design; rather, the problem lies with system users who are not skilled in cartographic design and who would probably never produce a map by conventional means, but would contract a cartographer to produce it.

It is unlikely that the general level of cartographic education of most computer map authors will be greatly increased, therefore cartographers must strive to make the programs used by naive map authors better able to produce reasonably well designed maps, or at least maps which do not break the most fundamental rules of map design.

The area of computer science devoted to producing programs that appear to respond intelligently to varied situations is Artificial Intelligence (AI). More specifically, an Expert System is a program which includes knowledge of how an expert solves a problem. It is essentially a program which includes a codified form of the rules that an expert uses to solve a problem, thus a cartographic design expert system would include the rules a cartographer uses when designing a map.

A long term goal would be to have a cartographic design expert system that could design any map at any scale, but current literature on expert systems suggests that at this time practical expert systems should be limited to narrow domains, ie. the problem area must defined within quite narrow margins. Some cartographic expert systems are currently under development. These have tended to concentrate on elements of the map or map design, eg name placement, line generalisation, solution of spatial conflicts, and while all these problems will have to be solved in any realistic production system, there would seem to be a need for the application of expert system techniques to more general design issues such as data selection, choosing an appropriate method of portraying a data set in map context, and generally trying to prevent the author from making poor design decisions when making the map.

ARTIFICIAL INTELLIGENCE

Artificial Intelligence is a branch of computer science involved in studying mental faculties and reproducing them through the use of computational models. The use of the word 'intelligence' may in fact be misleading as the term tends to be used to refer to individuals with above average creativity or cleverness, whereas most problems in Artificial Intelligence arise in attempting to recreate the mental capability of 'ordinary people'. AI is concerned with the general behaviour that goes along with intelligence; it is not limited to one particular method of producing 'intelligence', and the methods used may not be the same as people use (Charniak & McDermott, 1985; 7).

The ultimate goal of AI is to produce human-like intelligence in a non-human machine. Whether or not this is achievable does not reduce the importance of developing programs that take us towards that goal. The divisions of AI research can be seen as the elements to be solved in producing such a machine. While there is no universal agreement on the subdivisions of AI, the major groupings are Expert Systems, Natural Language Processing, Pattern Recognition and Robotics. Other common sub-headings are Computer Vision and Machine Learning, although the former of these is frequently encompassed by Pattern Recognition and the latter is really an essential component of any system which claims to have artificial intelligence.

David Forrest is currently Senior Lecturer, Department of Geography, Portsmouth Polytechnic. From 1st September: Lecturer in Cartography, Department of Geography and Topographic Science, University of Glasgow, Glasgow G12 8QQ.

While the general aim of AI research has not changed over its twenty-odd year history, that is to produce programs that can in some way 'think', there has been a shift in emphasis from trying to find general methods for solving a broad range of problems to that of solving very specific problems with very highly specialised programs.

Intelligence

The idea of intelligence is not concerned solely with what can be done, but also how it is done (i.e. the style or manner). The idea of intelligence is not concerned solely with what can be done, but also how it is done (i.e. the style or manner). Rather than rejecting a problem totally, or 'crashing', an intelligent system should degrade gracefully and provide a solution even if this is less reliable or complete (the user should be informed of this). Problems should be solved using insight and understanding rather than blind mechanical execution of rules and inference procedures should be used to answer 'what if..' type of questions, or to make predictions. Where conflicting options are possible, rather than using pre-assigned priority measures, the reasons for the conflict should be analyzed and the consequences of alternative choices should be considered.

There is a very thin division between programs that are clever and those that show artificial intelligence. Indeed, "... it is possible that there is no such thing as an intelligent program - just clever programs that become increasingly clever" (James, 1984; 116). It has been shown that by applying some simple rules one can give an impression of intelligence that would convince an innocent onlooker, but deeper investigation will ultimately reveal the lack of 'intelligence' of such systems.

Problem Solving

All branches of AI rely upon problem solving, to which there are two elements: Representation and Search. All of the approaches to problem solving require some sort of search for a solution. Conducting these searches as efficiently as possible is one of the aims of AI. However, before a search process can begin, the problem must be 'set up', or, in other words, a representation of the problem must be formulated.

> Usually one applauds a human problem solver not for conducting a fast and orderly search through all solution possibilities, but for looking at the problem in such a clever way that the solution seems elegantly simple.
>
> (Nilsson, 1971; 8)

There will often be alternative representations for the same problem, but unfortunately AI research is still directed at producing a generalised automatic method for the skilful formulation of problem representation.

Representation

The 'language' produced or operated upon during problem solving is known as the Internal Representation. This representation is, to some extent at least, an abstraction. The same representation may be embodied in a variety of different data structures, to make different operations efficient. It is normally assumed that it is easy to translate from one internal representation to another, and certainly easier than translation to and from external representations (i.e. questions and answers in English).

The internal representation is used by an AI program in the following way:

- When a program gets a statement, it translates it into an internal representation and stores it away.
- When it gets a question, it translates it into an internal representation as well.
- It uses the internal representation of the question to fetch statements from its memory.
- It translates the answer back into English.

(Charniak & McDermott, 1985; 11)

While this may seem more complex than simply storing the English, it is in fact more how people do things, in that we tend to remember the 'gist' of what we are told, long after we have forgotten the exact words. Specific knowledge representation methods for expert systems are beyond the scope of this article.

Search

AI programs work by searching the internal representation of knowledge for a solution, often referred to as a goal or 'the goal state'. In human intelligence we can see the parallel to this as being a specific response to solve a particular problem. Our reactions to certain situations may appear to be automatic, but are the result of all our thought processes being directed to achieve a certain 'goal'. We don't do things because we think, we think because we have things to do.

Typically the internal representation of a problem can be expressed as a tree structure or graph. This graph represents a structured series of nodes, each with an associated state descriptor. A solution is obtained by applying operators to these state descriptions until the 'goal state' is obtained (Nilsson, 1972). In the graph theory search process there are a number of ways in which the search can proceed. Basically these include 'blind search' procedures which always follow a predetermined pattern or 'best first' procedures which assess the path most likely to lead to the solution. If a system is to appear intelligent then intuitively this latter method is preferable. When a search proceeds down a blind alley, it must 'backtrack' to try an alternative route. Applications of the search problem to cartography have been discussed by Wilkinson (1987).

EXPERT SYSTEMS

Artificial intelligence may be described as the transfer of intelligence to machines. Expert systems deal with a small area of expertise that can be converted from human to artificial intelligence. Put simply, an expert system is a computer program which, by using facts and rules about a domain (problem), simulates the decision making process normally carried out by a human expert. They differ from conventional 'algorithmic' programs in both structure and operation.

It may seem that any computer program that solves a problem may be termed an expert system, but there are numerous points which distinguish an expert system from a conventional program, for example:

1. There is continuous interaction with the user, who conducts a dialogue with the system, and leaves with an answer or conclusion.

2. The system weighs up the likelihoods, explores alternatives and follows a course of reasoning which depends on the user's replies. Whole areas of investigation may be initiated or discarded as a consequence.

3. Uncertain or incomplete evidence is accepted and used.

4. The system elaborates on and explains why questions are asked, and describes how conclusions are reached.

5. Only significant questions are asked, and questions related to a particular topic are grouped together.

In a conventional program the user follows a rigorously defined series of steps to meet the requirements of the program exactly. In an expert system, the interaction is flexible and should emphasise the requirements of the user.

The Role of Expert Systems

An obvious question to ask is why there is a need for expert systems, rather than rely on human expertise. According to Basden (1984;61) the benefits lie in: greater reliability (will not forget factors); increased consistency (same importance given to factors); increased accessibility; the ability to arrive at a faster solution or try a greater number of alternatives in the time available; the easier duplication of expertise (less training). In the case of design, increased consistency also implies repeatability, something not always achieved in manual processes. It should also be easier to document and afford artificial expertise, and it is more permanent (Waterman 1986; 12). Expert systems are especially appropriate where there is no efficient algorithmic solution. "Such cases are called ill-structured problems ... " (Giarranto & Riley, 1989; 20), this typically being true of design problems.

There are of course disadvantages to expert systems, hence there is good reason not to eliminate human experts, but to supplement them. Human experts are creative and adaptive and although expert systems can gain through experience, they are not as flexible as humans.

EXPERT SYSTEM STRUCTURE

Although some of the superficial differences between expert systems and conventional programs have been discussed, it is perhaps in the underlying, internal structure that the differences are most apparent. The simplest model of an expert system consists of three main parts. These are the knowledge base, the inference mechanism or inference engine, and the user interface. Clearly an expert system for producing maps must also access a database of relevant data. It is the structural difference whereby the knowledge relating to the problem to be solved is separated from the inference mechanism that differentiates expert systems from conventional programs.

The term user interface is self explanatory, referring to the part of the system that communicates with the user. The skill, experience and judgement of one or more human experts is stored in the knowledge base. This can be viewed as a model of the experts' reasoning leading to one or more conclusions. The term knowledge is used by AI scientists to refer to the information a program needs before it can behave intelligently (Waterman, 1986; 16). This information generally takes the form of facts and rules about a particular topic or domain, although there a number of knowledge representation methods currently in use.

The inference engine is the component of the system which controls the order in which the knowledge base is used, generates new facts from existing rules and known facts, and is generally seen as the central module in an expert system. Expert systems work by relating the contents of the knowledge base to the information supplied by the user's answers to questions formulated by the system. The system infers the most appropriate action in any particular situation, either giving its solution, or asking further questions.

Heuristics

A traditional program is a list of instructions for giving a sure solution to a problem, or reporting that no solution exists. This is known as an algorithm. If one examines the way in which humans solve problems then one sees that very often an algorithm is not used, but a lose collection of 'rules of thumb' that seem to work are followed (James, 1984;10). While these rules often do not guarantee a solution, they make it more likely that you will get closer to one. Such rules of thumb are termed 'heuristics'. More explicitly, heuristics are: "... criteria, methods or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal. They represent compromises between two requirements: the need to make such criteria simple and, at the same time, the desire to see them discriminate correctly between good and bad choices" (Pearl, 1984; 3). Furthermore, when heuristics do produce a solution, it can take far less time than an algorithm would take for the same problem.

Heuristics are important in true expert systems because they are used to guide the search for a solution, ie they determine the best route to follow towards the desired result.

EXPERT SYSTEMS FOR WHOM?

Expert System can be used by a wide range of people for a variety of purposes. The major groups of users are likely to be experts themselves, practitioners[4], students and those with no experience in the field.

Experts will use expert systems in a decision support role, using them along with other decision support systems to confirm their decisions, or to act as intelligent checklists. The expert system may be used like an intelligent assistant and as more intelligence is added to it, it acts more and more like an expert. Also, although currently it does not appear likely that ES will replace the human specialist, they will reduce the number of trivial enquiries, thus allow the specialist to devote more time to less trivial problems.

Some expert systems will no doubt be used by novices, but there is some feeling that this will be less widespread than was at first thought. Most fields of study have their own special words or 'jargon' or apply special meaning to ordinary words that a novice might be dangerously unaware of (Basden, 1984; 64). Practitioners on the other hand will be familiar with the jargon of their domain. According to Basden it is also likely that expert systems for practitioners will be more cost effective than for novices.

Expert systems have much to offer in education because of the nature of expert systems generally and their capability to explain their reasoning. Students, like practitioners, will have some awareness of the 'jargon' and will be able to use expert systems in example cases, or to test their own hypotheses.

A final group who will likely make use of expert systems are specialists or experts in one particular domain wishing to apply their knowledge in a related field, or to make use of a system to process their information. For example, a geologist using a cartographic expert system to map his data. He has specialist knowledge about the geology of the area, but not the cartographic knowledge to produce the map. This use of expert systems receives little coverage in the literature. Perhaps the need for such systems for producing maps has something to do with how many potential map authors and developers of mapping systems view cartography. Most Computer Aided Design (CAD) systems are directed at assisting designers, not at making it easier for non designers to produce designs. Many computer mapping systems are intended for non cartographers to make maps.

As Robinson and Jackson state (1986; 431)

> Probably the user most at risk is the one who produces maps or other graphical output for his own use or for limited circulation. ... the user, in designing his output, will often use an interactive graphic facility and therefore he needs to optimise the information appearing on the screen appropriate to his particular expertise.
> ... [also] the final product may appear on a totally different medium, e.g. paper, which leads to further problems.

EXPERT SYSTEMS FOR CARTOGRAPHY

There are large areas of cartographic design where expert systems currently have little to offer. For example, designing a topographic series is an event that takes place relatively rarely and the design adopted is likely to be used for many years, perhaps with some modifications through experience, changing requirements, or new production methods. Each series has to take into account many factors, many of which will not be the same as the topographic series required for a different country or different scale. The time scale for producing the design is likely to be relatively long and there will be opportunities to consult acknowledged experts and experienced map users.

Similarly, the design of products such as hydrographic and aeronautical charts has evolved to a highly refined state and there are international agreements on many aspects of the design of such products.

In these cases expert systems will have many more benefits in production rather that design, particularly in many aspects of compilation where smaller scale maps have to be derived from larger scale (digital) source data. Many of the systems currently under development address these needs, such as those for line simplification, name placement, etc. Other likely areas of application for expert systems include the selection of soundings for hydrographic charts, contour interpolation, contour labelling, selective omission of minor tributaries in river networks, etc.

The two situations where expert systems do have much to offer map design are (a) where the map is likely to be a 'one off' design and is wanted quickly and (b) where the map author is not an experienced cartographer and cannot 'imagine' what the final design will look like. In many cases both of these circumstances will apply.

A broad overview of possible applications of expert systems to "cartographic processes" is provided by Granklanoff (1985). He also tries to assess quantitatively the suitability of expert systems for various mapping tasks. Each of 17 mapping tasks from geodetic control to printing were assessed for their suitability by eight mapping experts using standard criteria. Although a very limited study, several tasks relating to design feature near the top of the list for suitability, including generalisation and symbolisation, and feature selection and placement (Table I).

Table I

SUITABILITY OF TASK FOR APPLYING EXPERT SYSTEMS

Rank	Task Name
1	Source Evaluation
2	Source Selection & Compilation Planning
3	Generalization and Symbolization
4	Feature Selection and Placement
5	Stereo Photogrammetric Plotting
6	Typesetting & Type Placement
7	Geodetic Control Identification
8	Color Separation Proofing
9	Overlay Proofing
10	Analytical Triangulation
11	Mensuration
12	Distribution & Shipping
13	Inventory & Stockage Control
14	Press Printing
15	Engraving (Scribing)
16	Plate Making
17	Negative Preparation

After Granklanoff, 1985, p.621

Robinson and Jackson (1986) also identify a number of broad areas of cartography and digital mapping where expert systems could be of benefit. Their list includes: Manual and Automated Map Design; Digital Data-base/User Interface; Cartographic Education and Training; Spatial Data Error-train Analysis; Data Capture and Storage Standards; Data Format and Transfer Standards; and Replacing Cartographers. This is a very wide ranging list encompassing most areas of cartography, although they see the last entry on the list as being impractical for several reasons, not the least being the need for cartographers to provide their knowledge and monitor the achievements of automated systems!

BUILDING CARTOGRAPHIC DESIGN EXPERT SYSTEMS

Cartographic design, like most design problems, is characterised as being an unstructured problem. In order to create a cartographic design expert system the first step should be to formalise the map design process, but like many other areas of design this has yet to be done. If this can be achieved then what initially appears to be an uncomputable problem can be at least partially solved if it is properly divided up.

This lack of formalism has not prevented cartographers from designing maps. Evidence in the form of published maps indicates that the practice of cartography is well known, even if the cartographers concerned have not started from a theoretical analysis of what they are doing[5].

Interest has been shown in formalising the map design process, although this is very limited when compared to say architectural design. Eastman (1987) attempted to develop a "graphic syntax" for map design directed at expert systems applications, but failed to relate this directly to the actual process of designing maps. Mackaness and Scott (1987) attempted to define map design for expert systems. Aspects considered extend well beyond what might be considered the design process and discuss geographical knowledge and spatial cognition. They concluded that there is a wide range of aspects related to map design that need to be researched before any reasonable attempt can be made to automate the process, although they dismiss the notion of using expert systems to produce derived maps as 'simple', and concentrate on the processes involved in making the 'original' map. Forrest (1990, 1991) has also proposed a formalised map design process, in part based upon the relationship between the phenomena to be mapped, the data available, and the possibilities for representation.

The pessimistic view expressed by Mackaness and Scott and by many cartographers when expert systems are proposed seems to stem from a lack of understanding of both the map design process and expert systems. The apparent lack of written rules for cartographic design only causes concern if one considers the extreme range of possibilities for map types and map scales. Once the scale range, location, subject and purpose have been established, the options are greatly reduced and there are many examples of what can be done. That is, by moving from some vague notion of map design to the design of a specific map the problem becomes potentially solvable.

Thus, before attempting to develop a working system an attempt must be made to formalise the map design process. This should be based upon an understanding of the information to be mapped and the processes involved in producing a map. Once such a model has been established - and there may be several viable alternative models - progress can be made by codifying relevant cartographic practice. This may initially be in the form of verbal descriptions of the rules, conventions and processes required and then more formally in an appropriate structure to form an experts system's knowledge base.

CONCLUSION

Apart from a few very specialised aspects of design why are there still no cartographic expert systems? Proper development of expert systems needs a great deal of time, with substantive working systems often being quoted in 10s of man years. Academics don't have enough time and software developers currently don't have the economic justification to devote the time. Apparently, according to Mark Monmonier in 1990, Intergraph have invested in expert systems for GIS and currently have over assembled 6,500 rules.

Many of the expert systems that have been proposed for Computer Aided Cartographic Design have been inherently impractical. More than one has suggested that a single system will be capable of producing all manner of maps from large scale plans to world inventory maps, hydrographic charts to multivariate statistical maps, etc. This may come in time, but current thinking on expert systems is to limit the problem to 'narrow domains.' It is evident however that as long as expectations are realistic, artificial intelligence and expert systems do have much to offer cartography.

NOTES

1. Numerous terms have been used to describe maps produced with the aid of computers: computer cartography, computer mapping, computer assisted cartography, digital mapping, automation, etc. Here we are examining the design and production of maps for display on computer monitors (VDUs) or output on small format plotters and printer/plotters. The term **Computer Aided Cartographic Design** (cf. Computer Aided Design) will be used to refer to computers being used for the design and display of maps, whereas **Computer Mapping** will be used to refer to the broader use of computers in map making.

2. A Geographic Information System (GIS) should include a database of geographic information, tools for analyzing this data and the capability of mapping the results. The third part of such a system could usefully apply Computer Aided Cartographic Design.

3. The **Map Author** is one who conceives the map and prepares any special topic data. He may then proceed to carry out the design and production, or pass this on to the **Cartographer**. The **System User** is the user of a Computer Mapping system. He may be the Map Author and/or the Cartographer. The **Map User** may or may not be the Map Author or System User. The map's intended audience and purpose will have an influence on its design and production.

4. A practitioner is one who has some experience in a domain, but does not have the deep specialist understanding of an expert.

5. Keates, J.S. 1990 - pers. comm.

References

Basden, A. (1984) On the Application of Expert Systems. In Coombs, M.J. (ed.), *Developments in Expert Systems.* London: Academic Press Inc. (London) Ltd. pp. 59-75.

Charniak, E., McDermott, D. (1985) *Introduction to Artificial Intelligence.* Reading, MA.: Addison Wesley Publishing Company.

Eastman, J.R. (1987) *Graphic Syntax and Expert Systems Technical Papers,* ACSM-ASP Annual Convention. pp. 87-96.

Forrest, D. (1990) A Model of Cartographic Design for Expert System Applications. *Proceedings, Fourth International Symposium on Spatial Data Handling.* pp. 752-761.

Forrest, D. (1991) Classifying Cartographic Representations for Cartographic Design Expert Systems. *Proceedings, ICA91.*

GiarrantoJ., Riley, G. (1989) *Expert Systems: Principles and Programming.* Boston: PWS-Kent Publishing Company.

Graklanoff, G.J. (1985) Expert System Technology Applied to Cartographic Processes. *Proceedings, ACSM/ASP Fall Technical Meeting.* pp. 613-624

James, M. (1984) *Artificial Intelligence in Basic.* Sevenoakes, Kent: Butterworth and Co.(Publishers) Ltd.

Mackaness, W.A., Scott, D.J. (1987) The Problems of Operationally Defining the Map Design Process for Cartographic Expert Systems. *Research Paper RR-87-06,* School of Geography, Kingston Polytechnic.

Nilsson, N.J. (1971) *Problem Solving Methods in Artificial Intelligence.* New York: McGraw-Hill.

Pearl, J. (1984) *Heuristics.* Reading, Mass.: Addison-Wesley Publishing Company.

Robinson, G., Jackson, M. (1986) Expert Systems in Map Design. *Proceedings, Auto Carto 7.* pp. 430-439.

Waterman, D.A. (1986) *A Guide to Expert Systems.* Reading, Mass.: Addison-Wesley Publishing Company.

Wilkinson, G.G. (1987) The search problem in automated map design. *Cartographic Journal* Vol.24, No.1. pp. 53-55.

Selected Bibliography on Cartographic Expert Systems

Bouille, F. (1984) Architecture of a Geographic Structured Expert System. *Proceedings, International Symposium on Spatial Data Handling.* pp. 520-543.

Bouille, F. (1984) A Structured Expert System for Cartography Based Upon the HBDS. *Proceedings, Auto Carto Six,* Volume 2. pp. 202-210.

Bouille, F. (1986) Interfacing Cartographic Knowledge Structures and Robotics. *Proceedings, Auto Carto London.* pp. 563-571.

Bouille, F. (1988) Developing Strategies in GIS by problem-solving methods based on a structured expert system. *Proceedings, Eurocarto Seven.* pp. 42-50.

Broome, F.R. (1987) Automated Map Inset Determination. *Proceedings, AutoCarto 8.* pp. 466-470.

Chen, G. (1986) A Rule-Based Approach for Spatial Object Modelling and Task Management. *Proceedings, Auto Carto London.* pp. 588-597.

Cook, A.C., Jones, C.B. (1990) A Prolog Interface to a Cartographic Database for Name Placement. *Proceedings, Fourth International Symposium on Spatial Data Handling.* pp. 701-710.

Cook, A.C., Jones, C.B. (1990) A Prolog Rule-Based System for Cartographic Name Placement. *Computer Graphics Forum* 9 pp. 109-126.

Essinger, R. (1986) The Philosophy and Requirements of Computer-Aided Graphic Design in Cartography. *Proceedings, Auto Carto London.* pp. 189-196.

Fairchild, D. (1987) The Display of Boundary Information: A Challenge in Map Design in an Automated Production System. *Proceedings, AutoCarto 8.* pp. 456-465

Fisher, P.F., Mackaness, W.A. (1987) Are Cartographic Expert Systems Possible? *Proceedings, AutoCarto 8.* pp. 530-534.

Frank, A.U. (1982) MAPQUERY: Data Base Query Language for Retrieval of Geometric Data and Their Graphical Representation. *Computer Graphics.* Vol.16, No.3. pp. 199-207.

Heivly, C.G. (1986) Using Expert Systems Concepts to Fix USGS Digital Boundaries. *Proceedings, 2nd. International Symposium on Spatial Data Handling.* pp. 572-582.

Illert, A. (1988) Automatic recognition of texts and symbols in scanned maps. *Proceedings, Eurocarto Seven.* pp. 32-41.

Jaakkola, O., Sarjakoski, T., Blom, T., Laurema, M. (1990) From Satellite Data to Thematic Representation - A Knowledge-Based System for Cartographic Visualisation. *Proceedings, Fourth International Symposium on Spatial Data Handling.* pp. 711-720.

Jankowski, P., Nyerges, T.L. (1989) Design Considerations for MaPKBS - Map Projection Knowledge-Based System. *American Cartographer* Vol.16, No.1. pp. 85-89.

Jong, W.M. de, Wel, F.J.M. van der, (1990) Embedded Artificial Intelligence and Spatial Data Handling, Some Research and Prototyping Experiences. Proceedings, *Fourth International Symposium on Spatial Data Handling.* pp. 723-731.

Keates, J.S. (1989) Expert Systems and Cartographic Design. *Unpublished paper,* University of Glasgow. (Read to SC Summer School, Portsmouth 1990)

Kottenstein, T. (1990) Concept and Prototype of a Symbol Reference System for the Production of Thematic Maps. *Proceedings, Fourth International Symposium on Spatial Data Handling.* pp. 772-781.

Mackaness, W.A. (1986) Detection and Heuristic Resolution of Spatial Conflicts in Digital Map Datasets. *Unpublished manuscript,* Kingston Polytechnic, Surrey.

Mackaness, W.A., Fisher, P.F. (1987) Automatic Recognition and Resolution of Spatial Conflicts in Cartographic Symbolisation. *Proceedings, AutoCarto 8*. pp. 709-718.

Mackaness, W.A., Fisher, P.F., Wilkinson, G.G. (1985) The Design of a Cartographic Expert System. *Final Report for the Natural Environment Research Council*

Mackaness, W.A., Fisher, P.F., Wilkinson, G.G. (1986) Towards a Cartographic Expert System. *Proceedings, Auto Carto London*. pp. 578-587.

Maggio, R.C. (1987) The Role of Geographic Information Systems in the Expert System. *Proceedings, GIS 87 - 2nd Annual International Conference on GIS*. ASPRS/ACSM pp. 685-692.

Mark, D.M. (1986) Knowledge-Based Approaches for Contour-to-Grid Interpolation. *Proceedings, 2nd. International Symposium on Spatial Data Handling*. pp. 225-234.

Monmonier, M.S. (1986) Towards a Practicable Model of Cartographic Generalisation. *Proceedings, Auto Carto London*, Vol.2. pp. 257-266

Muller, J-C. (1990) Rule-Based Generalization: Potentials and Impediments. *Proceedings, Fourth International Symposium on Spatial Data Handling*. pp. 317-334.

Muller, J-C., Johnson, R.D., Vanzella, L.R. (1986) A Knowledge-Based Approach for Developing Cartographic Expertise. *Proceedings, 2nd. International Symposium on Spatial Data Handling*. pp. 557-571.

Muller, J-C., Zeshen, W. (1990) A Knowledge Based System for Cartographic Symbol Design. *Cartographic Journal*. Vol.27,No.1. pp. 24-30.

Nickerson, E.G., Freeman, H. (1986) Development of a Rule-Based System for Automatic Map Generalization. *Proceedings, 2nd. International Symposium on Spatial Data Handling*. pp. 537-556.

Nyerges, T.L., Jankowski, P. (1989) A Knowledge Base for Map Projection Selection. *American Cartographer*. Vol.16, No.2. pp. 29-38.

O'Callaghan, J.F., Robertson, P.K. (1986) Colour Image Display of Geographic Data Sets Using Uniform Colour Spaces. *Proceedings, 2nd International Symposium on Spatial Data Handling*. pp. 322-326.

Palmer, B. (1984) Symbolic Feature Analysis and Expert Systems. *Proceedings, International Symposium on Spatial Data Handling*. pp. 465-478.

Pfefferkorn, C. et. al. (1985) ACES: A Cartographic Expert System. *Proceedings, Auto Carto 7*. pp. 399-407

Poiker, T.K., Squirrell, R., Xie, S. (1982) The Use of Computer Science and Artificial Intelligence in Cartographic Design. *Proceedings, Auto Carto V*.

Roberston, P.K. (1988) Choosing Data Representations for the Effective Visualisation of Spatial Data. *Proceedings, Third International Symposium on Spatial Data Handling*. pp. 243-252

Robinson, G.J., Zaltash, A. (1989) Applications of Expert Systems to Topographic Map Generalisation. *Proceedings, AGI89*.

Roubal, J., Poiker, T.K. (1986) Automated Contour Labelling and the Contour Tree. *Unpublished paper*, Simon Fraser University.

Samson, L., Poiker, T.K. (1985) Graphic Design with Color Using a Knowledge Base. *Paper presented at the 10th CCA Annual Conference*.

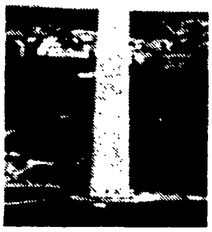
Trigg, A.D., Gill, G.A. (1988) Canvas: A System for Improved Colour Selection for Classified Imagery and Thematic Maps. *Report No.7*, NUTIS, Reading.

Vicars, D.W., Robinson, G.J. (1986) Generalisation from Large to Medium and Small Scale Ordnance Survey Maps Using Expert Systems Techniques. *Proceedings, Auto Carto London*, Vol.2. pp. 267-275.

Wilkinson, G.G., Fisher, P.F. (1987) Recent Developments in Geo-Information Systems. *Cartographic Journal* Vol.24, No.1. pp. 64-70.

Zeshen, W. (1990) A Representation Schema for Cartographic Information. *Proceedings, Fourth International Symposium on Spatial Data Handling*. pp. 782-791

Zoraster, S. (1987) Practical Experience with a Map Label Placement Program. *Proceedings, AutoCarto 8*. pp. 701-708.



Classifying Cartographic Representations for Cartographic Design Expert Systems

David Forrest
Department of Geography
Portsmouth Polytechnic
Portsmouth PO1 3HE
United Kingdom.

ABSTRACT

The application of expert systems to cartographic design is currently attracting considerable attention. In order to develop such expert systems the terms and processes involved must first be formally defined. This paper concentrates on the problems of assigning suitable methods of representation to the information to be displayed on a map prior to establishing more detailed specifications of the symbols themselves. In particular, the most commonly used cartographic representations are defined, and principles for their selection presented.

INTRODUCTION

Most cartographic text books distinguish between point, line and area symbols, and discuss various types of representation or 'map' such as the choropleth map, the dot map, etc. There appears, however, to be little organisation or classification applied to the various types of representation that may be used, and indeed for some methods there is no universally accepted name or descriptive term. If cartographic expert systems are to be developed in a rigorous fashion, then each type of representation must be clearly defined, as must the relationship it bears to the geographic phenomena and data it is representing.

For each representation method there will be a number of possible questions or variables that must be resolved before the specific symbols to be used can be specified. The development of these rules must take into account a wide range of possibilities and permutations depending on the graphic capabilities of the system, the type of map being made, the intended map use and map user, the priority of the category of information to be displayed, etc.

CARTOGRAPHIC REPRESENTATION

Despite the wide range of phenomena that may be mapped, there is a limited number of cartographic representations available. Table 1 outlines these. Some difficulty has been experienced in naming many of the representations, even those which are widely used such as **2C Categorical**. Often maps are referred to by the topic they cover, rather than the method of depiction used. There are exceptions: terms such as choropleth map and dot map are frequently used, which describe the method but not the data. Often the use of terms relating to the representational method used is inappropriate, because even on many simple maps several phenomena are depicted using a variety of representations.

Table 1. CARTOGRAPHIC REPRESENTATIONS

POINT SYMBOLS

0A **Dot Distribution.** All points have same symbol.
a represents occurrences of discrete individuals
b represents some fixed quantity grouped for each dot

0B **Categorised.** A range of features depicted by a set of point symbols of visually equal importance (e.g. tourist information symbols).

0C **Ranked.** Some ordinal ranking is implied by the range of symbols used.

0D **Proportional (graduated).** Symbols scaled according to some value or in fixed classes. May represent points, areas considered as points at map scale or areas represented at point location (usually zone centroid)
a classed unipolar distribution
b unclassed unipolar distribution
c bipolar distribution

0E **Subdivided Quantitative.** Symbol scaled to some value or fixed. May represent points, areas considered as points at map scale, or areas (usually by zone centroid).
a Fixed size
b Proportional

0F **Spot values.** Series of point locations with values. May have regular or irregular distribution.
a sparse locations, e.g. spot heights
b dense irregularly spaced, e.g. soundings, triangular irregular network
c regularly spaced, e.g. grid digital terrain model.

LINE SYMBOLS

1A **Boundaries.** Can represent:
natural boundaries (often zone of change)
actual ground feature (often man made)
intangible (no real existence on the ground)
a equivalence implied by symbols.
b hierarchy implied by line gauges or styles - typical of political/administrative boundaries.

1B **Networks.**
a network structure, e.g. roads. Hierarchy may be implied by symbols
b tree structure (branching) hierarchical networks e.g. river systems.

1C **Isolines (contours).** Line of known or assumed numerical value. Divides surface into zones between two values.

1D **Flow lines.** Can be in form of network, or independent routes/movement.

Table 1 (contd)

1E **Unstructured line symbols.** Miscellaneous, non network, often isolated and/or discontinuous line symbols not included in 1A or 1B (e.g. fences, walls, geological faults, etc.)

AREA SYMBOLS - each zone homogeneous; changes occur at zone boundaries.

2A **Isolated areas.** Binary division of surface - special case of 2C.

2B **Unclassed.** e.g. 'Political' map - simplified case of 2C.
a single level, all symbols visually equivalent
b hierarchy of symbols e.g. Country/state/county

2C **Categorical.** (Chorochromatic, colour patch, or mosaic map) Data feature coded. e.g. Land use, vegetation, geology.
a uniform
b hierarchical (i.e. sub-classes)

2D **Graded series.** Two possibilities for boundaries, but treatment of symbolisation the same:
Choropleth Delimited zones with numerical values - boundaries usually imposed on distribution (normally administrative, could be grid)
Dasymetric Delimited zones with numerical values - boundaries derived from distribution.
a unipolar variation of symbols
b bipolar variation of symbols
c bivariate symbolisation

2E **Layer coloured series.** Graphically the same as 2D but different spatial representation.
a unipolar variation of symbols
b bipolar variation (e.g. temperature zones)

2F **Hypsometric and Bathymetric colours.** Spatially the same as 2E, but difference in graphical treatment.

2G **Cartograms.** Areas scaled by a variable, not true extent.

SURFACE SYMBOLS - differ from areas in that continuous variation is represented.

3A **Shading.** (e.g. hill shading)

3B **Block Diagrams.** (2 1/2 D views) (Arguably not a true map - non orthogonal)
a continuous
b stepped

3C **3D Surface Model.** (cannot be displayed in this form - really an internal representation in a computer or a physical model, however the visualisation of the 3D surface could be considered in this class)

PHENOMENA, DATA AND REPRESENTATION

Many of the writings on cartographic design expert systems immediately launch into the problems of symbolising the information. However, in a proper analysis of the map design process one must first of all establish some basic facts about the information to be mapped. Arguably, one should go even further back in the process and examine the reasons why the map is to be made in the first place, its intended user or uses and many other factors. Apart from a few very basic questions this is beyond the scope of the present study, however a full consideration of the nature of information which is to be mapped is appropriate.

There are three aspects to be considered in assigning an appropriate representation: the nature of the phenomena being mapped; the locational data available about the phenomena; the measurement of the characteristics of the phenomena. Unfortunately, there seems to be no comprehensive study of the relationships between these aspects.

Most authors consider three basic categories of phenomenon: points; lines and areas. To this, several add surfaces or volumes. Strictly speaking temporal variation and movement should also be considered, but generally these are omitted from the current study. Although this is an apparently simple classification of the myriad possible phenomena, close examination reveals this to be a satisfactory approach in the majority of cases. Fundamentally, however the most important distinction is between continuous phenomena and discontinuous phenomena and this should ideally be reflected in their representation. For clarity, the terms discrete, linear, specific area(s) and continuous surface are used to describe the distribution of phenomena to distinguish them from data or symbol types.

This spatial data for mapping will in some way, although not necessarily simply, relate to a phenomenon or phenomena which has in some way been measured or sampled. While there are four possible distributions of phenomena, spatial data can only exist in the form of points, lines, or areas. Areas must either be defined by their outlines, which implies there must be line data, or as some regular tessellation of the surface (e.g. grid cell data). Area boundaries may be the actual outline of the phenomena, or some imposed (often arbitrary) boundary. To use the computer data structure analogy, these are 0-dimensional, 1-dimensional or 2-dimensional data.

In addition to the locational characteristics of the information we can also classify its attributes. When information is gathered, measurement is the process of assigning a class or value to the observation. This attribute of a feature may be either numerical or not. It can also be seen that non-numerical attributes can either be simply describing differences in kind or types of features, or can indicate ranks or hierarchies of features. This characteristic of information is known as its level of measurement. The conventional classes are nominal, where there is a change in type or kind, ordinal where there is a ranking of the data, and interval/ratio where some numerical value has been measured or calculated. Combinations of these classes are possible for some phenomena. For example, power stations may be distinguished by nominally by their method of generation and on a ratio scale of their output capacity. Knowledge of the level of measurement together with the nature of the phenomenon and the spatial data type will allow one or more cartographic representations to be assigned to the information.

Table 2 illustrates the relationships between phenomena, data, level of measurement and cartographic representation. Frequently it will be possible to depict a data set by more than

Table 2
RELATIONSHIP BETWEEN PHENOMENA, DATA AND REPRESENTATION

Phenomenon Distribution	Locational Data Dimension	Level of Measurement	Possible Representation Methods
discrete	0	nominal	0B,0A
discrete	0	ordinal	0C
discrete	0	interval	0D,[1C,2E]
discrete	1	ord/int	1C,2E
discrete	2	ord/int	2D,0D,[0A,1C,2E]
linear	1	nominal	1A,1B
linear	1	ord/int	1B,1D
specific areas	0	interval	0D,[1C,2E]
specific areas	2	nominal	1A,2A,2B,2C
specific areas	2	ord/int	2D,0D,[1C,2E]
continuous surface	0	interval	0F,[1C,2E]
continuous surface	1	interval	1C,2E

notes. 2 dimensional data is in the form of boundaries (outlines)
[ ] - requires further processing or additional information.

one representation method, although once the purpose of the map has been determined and the other information to be included has been decided, the choice is often limited. Where there remains a choice, the map author may decide which representation to use or accept the default option. It is planned to develop an expert system to classify information the user may wish to add to a database. This would take the form of a relatively simple classification type of expert system, which could run independently, or as a sub-system of a cartographic design expert system or indeed any G.I.S.

SYMBOLISATION

Although an experienced cartographer could devise an infinite number of symbols, for the purposes of developing an expert system this range must somehow be limited. Guidelines for the application of many of the representation methods outlined in Table 1 have been developed. To date these guidelines have concentrated on symbol design for maps in the scale range of 1:2million to 1:15million, but many of the principles are general and could be extended to other scales. Further limitations are placed on symbol design by factor such as the hardware and software used to develop the expert system, in particular the resolution and range of colours available on the output device used. Rules to account for all these factors must be defined if a useful system is to be produced.

THE DEVELOPMENT OF A FRAME BASED CARTOGRAPHIC DESIGN EXPERT SYSTEM.

ABSTRACT

The continual drop in price and enhancement in performance of computer hardware has brought the possibility of computer mapping to a wider audience. This increase in the availability of computer mapping capabilities has lead to a rise in the number of potential map authors, but does not however appear to have lead to more widespread knowledge of cartographic design theory. As it is unlikely that the general level of cartographic education of most computer map authors will be greatly increased, cartographers must strive to make the programs used by naive map authors better able to produce reasonably well designed maps, or at least maps which do not break the most fundamental rules of map design. To enable such a system to be developed the basic map design process must be formalised. This is seen as a preamble to developing specific rules which are incorporated into a prototype cartographic design expert system. The system is written in PROLOG with an inference engine designed specifically for map design problems. It uses frames as the main method of knowledge representation. Several of these frames are illustrated.

INTRODUCTION

In the last twenty years or so there have been a great number of programs created to produce maps using a computer.[1] Most of the commonly available programs are for producing small scale statistical maps, however more recently there has been a significantly increased interest in Geographic Information Systems for producing a wider variety of maps at a broader range of scales. The continual drop in hardware prices has brought the possibility of computer mapping to a wider audience, particularly with the increased power of micro-computers now available.

The increase in the availability of computer mapping capabilities has lead to a great increase in map authors[2] and the number of maps being produced, but does not however appear to have lead to more widespread knowledge of cartographic design theory. The large number of poorly designed maps created by map authors using computer systems to produce their own maps indicates that there is a lack of knowledge of how to design maps. These poorly designed maps are not the fault of the computer programs. Most programs do have the capability of producing well designed maps when used by someone knowledgeable in map design. The problem lies with authors who are not skilled in cartographic design and who would probably never produce a map by conventional means, but would contract a cartographer to produce it.

It is unlikely that the general level of cartographic education of most computer map authors can be greatly increased, nor than computer map design can be limited to those with cartographic training, therefore cartographers must strive to make the programs used by non cartographers better able to produce reasonably well designed maps, or at least maps which do not break the most fundamental rules of map design. The area of computer science devoted to producing programs that include knowledge of how an expert solves a problem is that of Expert Systems. An Expert System is essentially a program which includes a codified form of the rules that an expert uses to solve a problem, thus a cartographic design expert system would include the rules a cartographer uses when designing a map.

A long term goal would be to have a cartographic design expert system that could design any map at any scale, but current literature on expert systems suggests that at this time practical expert systems should be limited to narrow domains, ie. the problem area must be well defined within quite narrow margins. Several cartographic expert systems are currently under development. These have tended to concentrate on elements of the map or map design, eg name placement, line generalisation, solution of spatial conflicts, and while all these problems will have to be solved in any realistic production system, there would seem to be a need for the application of expert system techniques to more general design issues such as who the intended user is, why the map is required, what its intended use is, where it is to show, what information

[1] Numerous terms have been used to describe maps produced with the aid of computers: computer cartography, computer mapping, computer assisted cartography, digital mapping, automation, etc. In the current study the term Computer Aided Cartography (cf. Computer Aided Design) will be used to refer to computers being used for the design and display of maps, whereas Computer Mapping will be used to refer to the broader use of computers in map making.

[2] The **Map Author** is one who conceives the map and prepares the data. He may then proceed to carry out the design and production, or pass this on to the **Cartographer**. The **Map User** is the intended audience of the map. This may or may not include the Map Author. The **System User** is the user of a Computer Mapping system. He may be the Map Author, the Cartographer, or the Map User.

is required, how to represent the information in map context, and generally trying to prevent the author from making poor design decisions when making the map.

Several papers on cartographic design expert systems seem to assume that the process starts with the symbolisation of the map information. Clearly an examination of the map design process shows that this comes relatively late in total process. The system described here commences with questions about the nature of the map and its intended purpose, establishing a background to the map. Discussions with cartographers have indicated that this could go further, or perhaps even back a step to question the map author as to why he wants a map in the first place and what he expects it to achieve. While this approach may be desirable, until we have much more sophisticated natural language interpreters it is unlikely that this will be achievable in any realistic sense with a flexible form of dialogue.

CHOICE OF SUBJECT

As stated in the introduction, current wisdom on developing practical expert systems suggest that relatively well defined narrow domains should be chosen. They should be applied to subjects where there are human experts who regularly perform the task better than most other people. "Designing an expert system to add single digit numbers is silly, because almost everyone does this well. On the other hand, designing an expert system to predict the stock market is doomed to failure as no human expert does this consistently well (Bahill & Ferrell, 1986; 50). Most expert systems have been developed for well structured problems which can be easily formalised. Design problems have proved especially difficult in this context and design has been characterised as an ill-structured problem, " ... i.e. one which is difficult to formalize and difficult to solve, especially with man made problem solvers" (Begg, 1984; 45).

In the initial stages of development, success will most likely be achieved if the expectations of what a system will produce are kept within limited bounds. Keates (1990) identifies three main ways in which maps are designed: accidental design; imitative design; and creative design. If the system is capable of the second of these and helps to prevent poor examples of the first, then it must be judged as being successful.

In order to create a cartographic design expert system the first step must be to formalise the map design process, but like many other areas of design this has yet to be done. If this can be achieved then what initially appears to be an uncomputable problem can be at least partially solved if it is properly divided up.

In identifying a narrow domain within cartographic design there are two limiting factors to be considerd, map type and scale. If the map types which can be produced are limited to a single type or small group of related types, e.g. geological maps, or climatic maps, then specific rules for these maps at a wide range of scales could be developed. If a broader range of map types is desired, then to develop a practicable system the range of map scales considered must be limited. This limitation on scales is imposed due to problems of generalisation, many aspects of which have still to be automated satisfactorily.

Thus, to formalise the cartographic design process the type of maps and an appropriate scale range to be examined must be defined. The map types selected for this project are those of individual countries, parts of countries or groups of small countries found in regional atlases, such as those used for secondary education. An example would be the home country and regional maps in products like "A Senior Secondary Atlas for Nigeria" by Collins-Longman (1983). This type of atlas, while being in part a general world reference atlas, includes a number of maps devoted to a variety of special topics of the home country and surrounding region. Similar maps are also found in textbooks and atlases about specific countries and regions.

Scale for these maps typically ranges from about 1:2000000 (1 : 2 million) to 1:15000000 (1 : 15 million). The format of this type of product is also appropriate given the hardware limitations of most micro computer systems. These atlases are typically A4 size (about 30 x 20 cm) or smaller, therefore maps larger than A3 size do not have to be considered. A common size for computer monitor displays is around 30 cm wide by 20 cm high. Thus, maps intended for hard copy output up to this size can be shown true to scale on the display; an A4 upright image would be shown at about 75% of its true size and an A3 downright image at about 50% of its true size. Other factors have to be taken into account, but it is at least practical to consider displaying these maps on typical monitors, and also to produce hard copy on inexpensive printers and plotters. Designing maps specifically for slides and overhead projection foils would be obvious extensions. For the prototype system however, only maps displayed on the computer monitor will be considered in detail.

In addition to small scale topographic maps, the type of atlas mentioned have a wide range of special topic maps, such as relief maps, land use, communication, climate, etc, which include point, line and area data in both numerical and non-numerical form, thus rules will be required to deal with most types of information found on maps. It is intended that a database containing base data and special topic data will be part of the system. This will allow a variety of maps to be produced. Apart from this database, the map author must also be able to add special topic information for the system to be of more general use. The system must be capable of interacting with the author to allow him to describe the nature of the phenomena to be mapped and the information available.

The remainder of this paper falls into three sections. The first is a consideration of information for mapping and its representation; the second details the basic map design process and how it may be automated; and the third outlines the protitype sytem and illustrates some of the frames used for knowledge representation. In some cases broad generalities are stated. These are not intended to be specific solutions to all situations that may occur in cartographic design, but descriptions of some of the problems that may face the proposed system and their possible solution. This is seen as a preamble to developing specific rules as the system develops.

PHENOMENA, DATA AND REPRESENTATION

There are four aspects to be considered here: the nature of the phenomena being mapped; the locational data available about the phenomena; the measurement of the characteristics of the phenomena; and its possible cartographic representations.

Many of the writings on cartographic design expert systems immediately launch into the problems of symbolising the information. However, in a proper analysis of the map design process one must first of all establish some basic facts about the information to be mapped. Arguably, one should go even further back in the process and examine the reasons why the map is to be made in the first place, its intended user or uses and many other factors. Apart from a few very basic questions this is beyond the scope of the present study, however a full consideration of the nature of information which is to be mapped is appropriate.

Characteristics of Phenomena

Most cartographic texts consider three basic categories of phenomenon: points; lines and areas. To this, several add surfaces or volumes. Strictly speaking temporal variation and movement should also be considered, but generally these will be omitted from this study. Although this is an apparently simple classification of the myriad possible phenomena, it is worth examining the four main classes in more detail. Fundamentally, however the most important distinction is between continuous phenomena and discontinuous phenomena and this should ideally be reflected in their representation.

A phenomenon distributed at points appears intuitively simple with each occurrence being denoted by a set of coordinates. However, very few features actually occur at a point in the strictest sense of the term. Normally what we are considering are discrete features of some finite size, which are discontinuous in their coverage and which we deem to be points given the scale of the map. For example a factory can cover some considerable number of square meters on the ground. On a large scale plan it is be represented as an area enclosed by its walls, but on small scale maps it may well be shown by a point symbol representing the existence of the factory or by a symbol representing the value of its output. Another classic example of this is the treatment of towns and cities on small scale maps. Clearly these spread over some considerable extent, but for mapping purposes we can consider them to be distributed at points.

Many discrete phenomena are seldom considered as occurring at points, such as population. "Generally they are not fixed in location, and can only be recorded as being present at or within a location at a given time." (Keates, 1989; 205) Rarely do we have data on individuals. Normally they are enumerated by some area and may be represented by area or point symbols depending upon map scale, purpose and design.

Linear phenomena by their very nature must be continuous, (in space if not in time) although often what we consider to be linear features are actually zones of transition between two surfaces, such as the coastline, and indeed most 'natural' boundaries are of this type. They may also be tangible features on the ground such as fences or walls, but again our treatment of many features will depend upon map scale, e.g. a road at large scale may be depicted by two bounding kerbs representing its physical extent, whereas at small scale it is its importance as a line of communication that is mapped.

Phenomena occurring within specific areas may be present continuously over the whole surface, such as geology, soils, etc., or may be discontinuous, such as lakes, built-up area, etc. Effectively the latter type is a subdivision of the first type where we have a binary division of the surface, rather than a more numerous set of classes and sub-classes.

Surfaces refer those phenomena which are continuouse and vary quantitatively, in space, the topographic surface being the obvious example. Some phenomena vary continuously both spatially and temporally, such as much climatic information. Precipitation, while not continuous, may also conveniently be grouped here (Keates, 1989; 205) as we are normally dealing with averages over time. Variables such as temperature and pressure if considered at some specific level, e.g. ground level may also be treated similarly.

For clarity, the terms discrete, linear, specific area(s) and continuous surface will be used subsequently to describe the distribution of phenomena to distinguish them from data or symbol types.

Relationship Between Phenomena and Locational Data

Before one can commence to design a map one must have data to map. This data will in some way, although not necessarily simply, relate to a phenomenon or phenomena which has in some way been measured or sampled. While a phenomenon may be distributed at discrete locations, along lines, contained in specific areas, or vary continuously over a surface, spatial data can only exist in the form of points, lines, or areas. Areas must either be defined by their outlines, which implies there must be line data, or as some regular tessellation of the surface (e.g. grid cell data). Area boundaries may be the actual outline of the phenomena, or some imposed (often arbitrary) boundary.

To use the computer data structure analogy, these are 0-dimensional, 1-dimensional or 2-dimensional data. Despite this seemingly simple classification of phenomena and data, frequently the data available about a phenomenon has not been collected for the purpose of mapping and may not reflect the actual distribution of it. This situation may also be compounded by the available data being of a secondary nature, having been preprocessed for a variety of reasons.

While there is an unlimited number of phenomena that may be mapped we can identify a small number of frequently recurring combinations of distributions of phenomena and the spatial data available about them. These are illustrated in Table 1, from which 8 primary combinations emerge:

1. Discrete units, specific (point) location data
2. Discrete units, data aggregated by bounded area
3. Discrete units, data aggregated by cells
4. Linear phenomenon, with line data
5. Specific areas, outline data
6. Specific areas, cell data
7. Continuous surface, data sampled at points
8. Continuous surface, data sampled along lines.

In addition to these primary combinations, there are several possible secondary or derived distributions. Frequently information about specific areas is attributed to a single point within the zone, usually its centroid. Any set of numerical point data may have isolines produced from it, thus we may have discrete units aggregated by specific areas, data attributed to a point within each unit, and isarithms interpolated from these. Other such complex permutations are possible, but there may be occasions when the map author may only know the nature of the phenomenon and the form of data available, but not how the data has been derived.

Table 1

RELATIONSHIP BETWEEN PHENOMENA AND LOCATIONAL DATA

Frequently Occurring Combinations.

Data Dimension	0	1	2 Bounded	2 Tessellation
Phenomena				
Discrete Units	1	2	1 or 2	1
Linear		1		
Specific Areas	2		1	1
Continuous Surface	1	1 or 2		

1 - primary data
2 - secondary or derived data

Level of Measurement

In addition to the locational characteristics of the information we can also classify its attributes. When information is gathered, measurement is the process of assigning a class or value to the observation. This attribute of a feature may be either numerical or not. It can also be seen that nonnumerical attributes can either be simply describing differences in kind or types of features, or can indicate ranks or hierarchies of features. This characteristic of information is known as its level of measurement (Robinson et al., 1984). The conventional classes are nominal, where there is a change in type or kind, ordinal where there is a ranking of the data, and interval/ratio where some numerical value has been measured or calculated and the range of possible values is continuous. Knowledge of the level of measurement together with the nature of the phenomenon and the spatial data type will allow one or more cartographic representations to be assigned to the information.

Cartographic Representation

Despite the wide range of phenomena that may be mapped, there is a limited number of cartographic representations available. Table 2 outlines these. Frequently it will be possible to depict a data set by more than one of these methods, although once the purpose of the map has been determined and the other information to be included has been decided, the choice is often limited. Where there remains a choice, the map author may decide which representation to use or accept the default option.

Table 3 illustrates the relationships between phenomena, data, level of measurement and cartographic representation.

Table 2

CARTOGRAPHIC REPRESENTATIONS

POINT SYMBOLS

0A **Dot Distribution.** All points have same symbol.

a represents occurances of discrete individuals

b represents some fixed quantity grouped for each dot

0B **Categorised.** A range of features depicted by a set of point symbols of visually equal importance.

0C **Ranked.** Some ordinal ranking is implied by the range of symbols used.

0D **Proportional (graduated).** Symbols scaled according to some value or in fixed classes. May represent points, areas considered as points at map scale or areas represented at point location (usuallly zone centroid)

a classed unipolar distribution

b unclassed unipolar distributions

c bipolar distribution

0E **Multivariate Quantitative.** Symbol scaled to some value or fixed. May represent points, areas considered as points at map scale, or areas (usually by zone centroid).

a Fixed size

b Proportional

0F **Spot values.** Series of point locations with values. May have regular or irregular distribution.

a sparce locations, e.g. spot heights

b dense irregularly spaced, e.g. soundings, triangluar irregular network

c regularly spaced, e.g. grid digital terrain model.

LINE SYMBOLS

1A **Boundaries.** Can represent:

natural boundaries (often zone of change)
actual ground feature (often man made)
intangible (no real existance on the ground)

a equivalence implied by symbols.

b hierarchy implied by line gauges or styles - typical of political/administative boundaries.

Table 2 contd.

1B **Networks.**
a network structure, e.g. roads. Hierarchy may be implied by symbols
b Tree structure (branching) hierarchical networks e.g. river systems.

1C **Isolines (contours).** Line of known or assumed numerical value. Divides surface into zones between two values.

1D **Flow lines.** Can be in form of network, or independant routes/movement

1F **Unstructured line symbols.** Miscellaneous, non network, often isolated and/or discontinuous line symbols not included in 1A or 1B (e.g. fences, walls, geological faults, etc.)

AREA SYMBOLS - each zone homogenious; changes occur at zone boundaries.

2A **Isolated areas.** Binary division of surface - special case of 2C.

2B **Unclassed.** e.g. 'Politcal' map - simplified case of 2C.
a single level, all symbols visually equivalent
b hierarchy of symbols e.g. Country/state/county

2C **Categorical**. (Chorochromatic, colour patch, or mosaic map) Data feature coded. e.g. Land use, vegetation, geology.
a uniform
b hierarchical (i.e. sub-classes)

2D **Graded series.** Two possibilities for boundaries, but treatment of symbolisation the same:

> **Choropleth** Delimited zones with numerical values - boundaries usually imposed on distribution (normally administrative, could be grid)
>
> **Dasymetric** Delimeted zones with numerical values - boundaries derived from distribution.

a unipolar variation of symbols
b bipolar variation of symbols
c bivariate symbolisation (not Dasymetric)

2E **Layer colours.** (e.g. hypsometric colours)
a unipolar variation of symbols
b bipolar variation (e.g. temperature zones)

2F **Cartograms.** Areas scaled by a variable, not true extent.

Table 2 contd.

SURFACE SYMBOLS - differ from areas in that continuous variation is represented.

3A **Shading.** (e.g. hill shading)

3B **Block Diagrams.** (2 1/2 D views) (Arguably not a true map - non orthogonal)
a continuous
b stepped

3C **3D Surface Model.** (cannot be displayed in this form - really an internal representation in a computer or a physical model, however the visuallisation of the 3D surface could be considered in this class)

It is planned to develop a separate expert system to classify information the user may wish to add to the database. This would take the form of a relatively simple classification type of expert system, which could run independently or as a sub-system of the main package.

THE DESIGN PROCESS

The general procedure for designing a map follows the route of: Compose; Compile; Symbol specification; Display; Adjust. A similar route can be followed with computer aided cartography, and indeed it is logical that an expert system follow a similar route to a Cartographic expert. The stages to be followed are: Description; Layout; Data Selection; Symbolisation; Display; Modify. These are described in more detail below.

Description

In this step the aim is for the user to describe to the system some basic information about the map he requires.

First then, the user must inform the system of the type of map to be produced. Immediately, a distinction can be made between topographic or 'general purpose' maps and special topic maps. A topographic map ideally shows all information with the same level of importance, i.e. no one aspect of the map should dominate, although in practice cultural information tends to dominate on most topographic maps, and in any good design there should in any case be several 'visual levels'. For a special topic map, the special topic information normally will be the dominant part of the graphic image with the base information providing context and orientation for the map user. Thus, the system will have to know what type of map is required at an early stage.

The system will 'know' about a definitive number of map types that can be produced from the information in its database, however the user may not be familiar

Table 3

RELATIONSHIP BETWEEN PHENOMENA, DATA AND REPRESENTATION

Phenomenon Distribution	Locational Data Dimension	Level of Measurement	Possible Representation Methods
discrete	0	nominal	0B,0A
discrete	0	ordinal	0C
discrete	0	interval	0D,[1C,2E]
discrete	1	ord/int	1C,2E
discrete	2	ord/int	2D,0D,[0A,1C,2E]
linear	1	nominal	1A,1B
linear	1	ord/int	1B,1D
specific areas	0	interval	0D,[1C,2E]
specific areas	2	nominal	1A,2A,2B,2C
specific areas	2	ord/int	2D,0D,[1C,2E]
continuous surface	0	interval	0F,[1C,2E]
continuous surface	1	interval	1C,2E

notes. 2 dimensional data is in the form of boundaries (outlines)
[ ] - requires further processing or additional information.

with such titles and may require help in defining what he wants. He may indeed require a map that the system does 'know ' about, but refer to it by a different title. From the author's description of the information to be included in the map the system should be able to determine the type of map required. The user's name for this may be added to the knowledge base for future reference.

If a topographic map is required then the system will be able to exercise almost total control over the map design as all the information will be contained in the system data base and the system 'knows' about this type of map. If a special topic map is required, more information will be needed from the user, particularly if the main information to be mapped is not included in the system data base. In this event the user will have to describe the phenomena to be mapped, the data available, and supply the necessary spatial and/or non spatial data.

The purpose for which the map is to be used is also important in determining what must or may be included in the map, the level of detail which may be required and hence the scale that will be required to show the desired detail. As the system is specifically limited to producing small scale maps in a particular scale range, there are obviously limits on the intended use of the output. Clearly maps at these scales are not intended for detailed measurement, but more for providing general information about an area or an overview of specific distributions in an area.

The level of detail required on the map will influence the amount of information selected and also the level of generalisation used. It is closely related to the map purpose and to the scale of the map, scale probably being the most important limiting factor in the amount of detail that can be shown.

The intended user of the output should also be considered at this stage. The map author can also be the map user, but the map may be being produced for a wider audience. If the author is also the intended user then one can normally assume some familiarity with the location or information being portrayed. If the map is intended for a wider audience they may or may not be knowledgeable about the area or subject. If the map is intended for a naive audience it may be desirable to make the map simpler than for an expert audience. This will reflect on both the content of the map and the level of detail at which each element is shown.

Layout

The factors to be decided at this stage are the actual geographical area to be mapped; the format of the output; and the scale of the output. A decision on the first of these and one other will determine the third. The level of detail determined above may influence the choice of scale. Some backtracking may be required if it is not possible to show the desired area in the available format at a scale commensurate with the topic or the level of detail requested.

Location. As a general point, the map author will know within reasonable limits the area to be mapped, although the size and shape of the area of primary interest and the purpose of the map may influence how much of the surrounding area should be included. It is unlikely that the system will be of much assistance in determining this, although an interim plot of the area may help the author.

Format. If the map is only to be displayed on a CRT display then the format will most likely be the maximum size allowed on the display. If hard copy is required this will be limited by size of printer or plotter to be used, but may also be limited by other factors, such as the page size of a publication. If the map is for reproduction the author may have fixed criteria. If the author needs help at this stage then some basic rules of appearance can be used, such as ratio of sides etc, and selection of portrait or landscape position depending on shape of area.

Scale. The general principle with atlas maps is to use the maximum scale possible (to the nearest round figure) within the given format, thus this will normally be calculated by the system based upon the size of the area and the format. It is possible that the user will require the map to be of a certain scale (if one of a series perhaps), thus once scale and location have been specified the required format can be calculated. Scale is a topic that is frequently misunderstood by map authors and users, particularly at the small scales used here so it is likely that the system will have to provide a considerable amount of explanation.

Marginal Information. The proposed system will not at this time make any attempt to design the layout of marginal information such as titles, legends, etc. Reserving space for this will influence the scale or format of the map, but will be the authors responsibility. Future developments could include assessment of the outline of the map area, the amount of legend space required, etc., and the suggestion of possibilities to the map author. Initially the title, scale and legend will be placed outside the map neat line, with the title and scale across the top of the map and the legend on the right hand side, if shown.

Data Selection

The information to be included in a given map will depend upon the type of map, the scale, and the level of detail required. Table 4 lists the information to be included in the system database. Space does not permit a fuller description of this data.

Topographic Maps. If a topographic map is required then, as all the information will be contained in the system database and the system will be 'familiar' with this type of map, the system will be able to exercise almost total control over the selection and representation aspects of design. All the information described as 'basic information' in Table 4 would normally be included, although the user could be given the choice of a 'physical' type map which includes hypsometric tints or a 'political' type map which has coloured administrative zones. The level of detail at which each of feature is depicted will however depend upon the scale, the purpose and the specified level of detail of the map.

Special Topic Maps. If a special topic map is being produced, more information will be needed from the user, particularly if the main information to be mapped is not included in the system data base. Maps whose topic is one of those listed as supplementary information in Table 4, while requiring more user input than topographic maps, will require considerably less than for special topic information supplied wholly by the map author. The basic cartographic representations of the information in Table 2 will be known to the system, as well as what base information is normally included in a map of the selected topic (i.e. there will be information about this in the knowledge base).

If information to be mapped is not in the database the user would be prompted to describe the phenomenon and the data he has available at this stage. This can in fact be seen as a separate task to the design aspects of the system and fits the model of a classification expert system.

While perhaps to be avoided for reasons of simplicity, it is possible that a map author will require a map showing more than one special topic. (The use of bivariate mapping of statistical/census information is not included in this discussion.) The map may for example show both temperature and precipitation. As there are strict limits on the number of continuous phenomena that can be shown by area symbolism -

Table 4

Proposed Database Contents.

BASIC INFORMATION

Political/Administrative Boundaries - International and internal (2 levels if available). These will be used with separate census data file for statistical maps.

Coastline - similar level in hierarchy as International boundaries. Should be able to be shown at two levels of generalisation.

Drainage - network classified with at least 3 levels.

Lakes - large lakes (say greater than $2mm^2$ at the largest map scale). Must be linked with drainage network.

Railways - one level.

Roads - classified as highways/motorways, major roads, other roads.

Settlements - (administrative definition) database would contain classification based on simple hierarchy, e.g. National capital, State capitals, other cities and important towns, giving 4 levels of hierarchy.
Ideally settlements would be chosen from a separate database with a variety of factors, e.g. population, political status, remoteness, etc., with a ranking calculated from these. Cutoff point determined by scale initially. User could specify number of settlements to be included, or selection criteria based on facts in database and system make choice. Different default parameters could be specified for different map types. Number of categories dependent on scale, number of settlements to be included and map type.

Contours - frequently shown on atlas maps with non uniform interval, as basis of layer colours. Database should include all contours based upon minimum interval appropriate for region. Actual intervals used selected automatically depending upon scale. Area symbolisation may be included depending on the type of map.

SUPPLEMENTARY INFORMATION

(Information frequently used for special topic maps in regional atlases)

Geology, Soils, Land Use, Land Cover (Vegetation), Annual Precipitation, Average Temperature.
Census data, including population etc. (to be used with administrative boundaries above)

probably two at most, one being shown by area colours and the other by area patterns - the user will have to prioritise the phenomena to be depicted, and may have to opt for line or point symbols to depict some continuous phenomena. For example, a map could show annual precipitation by layer tints and January and July isotherms by two sets of lines.

Base Information for Special Topic Maps. In designing a special topic map one must have an appropriate base map on which to display the information. There are two common approaches to this. The first is to take a topographic map and reduce it to a background image, often by printing it in grey. This will mean that much superfluous information will be included and also that some essential information may be obscure.

The second approach is to design the base specifically for the map. This involves selecting the appropriate information from the topographic base and symbolising it to compliment the special topic information. This should result in a better solution.

Frequently little consideration appears to be given to the level of detail of the base image when compared to the special topic representation. For example, a map with a very detailed coastline showing very generalised climatic information can mislead the user into thinking the special topic information is as detailed as the topographic information. Thus, some attempt must be made to have commensurate levels of detail for different elements of the map. This may involve simplifying the base information to reflect more closely the detail or accuracy of the special topic information.

Map complexity. Although the level of detail, map purpose and scale together will provide some indication of how much information should be included in the map, some problems will only emerge after the data has been (provisionally) selected. For example if a coverage of areas is to be included the system must check to see that the polygons are large enough to be perceived. If not, a more generalised representation must be used. Similar tests can be done on total length of line and number and average spacing of point symbols. Although this is not a true measure of complexity as it doesn't take distribution into account, it provides an initial indication to the system that the map may be too detailed.

Interaction of representations. It is generally not possible to show many sets of area symbolisation. Some areas may have to be implied by their boundaries. Problems of spatial conflicts have been deal with in some detail by Mackaness (1986) and solutions developed therein could be incorporated. Generally, and whenever possible, problems should be avoided by not selecting too many data sets.

Generalisation. This creates many problems in map design, particularly as scale decreases. There have been several studies on expert systems applied to map generalisation and it is beyond the scope of this study to incorporate all the possibilities. Given that the range of scales available to the system is limited, generalisation can be resolved partly by selection and, where appropriate, by having two sets of linear data or coding linear data so that it can be produced at two levels of generalisation, determined by scale and level of detail required.

Symbolisation

Having made an initial selection of the information to be included in the map (which may have to be modified once the map has been displayed) each element of the map will have to be given a symbol specification. The actual details of this will vary quite considerably depending on the phenomena being mapped, the scale, etc., however the first step is to assign the cartographic representation to be used (Table 2). Each of the data sets to be included in the data base may be assigned one or more possible representation based upon the nature of the phenomena, the locational data and its level of measurement as discussed above.

This is only the first step in specifying the symbols. Once the type of representation is known, specific symbols will have to be assigned to the information. In some cases this will be trivial, such as specifying colour and gauge of rivers. In other cases considerable effort will be required to select the most appropriate set of point symbols or area colour scheme, for example. Rules for the representation and for the data set will be used to narrow the choice, but inevitably user choice will play a major role here, or at least in approving defaults or choices suggested by the sytem.

Display

Having determined the information to be included and its graphical representation the map can be displayed on the screen. This is largely a procedural task for the system, although, due to the nature of computer graphics and some of the representation methods, consideration will have to be given to the order in which items are drawn. Generally speaking area symbols will be produced first followed by lines, then points. Text, although currently excluded, would be added last. More sophisticated measures will be required in many cases for hard copy output, in particular the masking of underlying symbolisation so that subsequent features are visible.

Modify

It would be ambitious to suppose that the first attempt at designing the map will be exactly what the user requires, thus the system must be able to interact with the user to modify any of the decisions previously made. Any modifications requested would of course have to be processed through the knowledge base, and the user notified of any knock on effects that may occur. The system will store parameters for completed designs so that it is possible to backtrack should the modification not result in an improved map.

The biggest difficulty here however, is in assessing what good design is, as this is largely subjective and attempts to quantify this (e.g. Mackaness et al., 1986) bear little resemblance to the user's perceptual response. It is also arguable that the intended user of a cartographic design expert system is unlikely to be able to pinpoint what the design problems are far less quantify them, therefor modifications are more likely to affect what is shown, rather than how it is shown. This is an area requiring considerable additional research.

PROTOTYPE EXPERT SYSTEM

In order to test the model of cartographic design and the rules formulated, a prototype map design expert system has been developed. The system is written in PDC Prolog (formerly Turbo Prolog). The use of existing shells was investigated, but deemed to be impractical due to the nature of the problem and its graphical requirements. This view is supported by others developing Intelligent CAD and map design systems (e.g. Ditterich & Ullman, 1987, Muller & Zeshen, 1990)

PDC Prolog is a 'typed' Prolog compiler which has the advantage of many built in graphics predicates allowing relatively easy development of graphics programs without the need to interface to other languages or systems. A major disadvantage is the limitation that only facts may be asserted into the knowledge base at run time. This makes it harder (but not impossible) for new rules, either inferred by the system or supplied by the user, to be added.

The inference method used is a predominantly forward chaining mechanism, i.e. it is a data driven solution, where rules are matched against facts to establish new facts. Backward chaining using hypothesis testing is used at some points, mainly to establish if a default value is the most appropriate. An overall backward chaining approach has been attempted for map design (e.g. Muller & Zeshen, 1990), but in the author's opinion the lack of adequate evaluation procedures for assessing map design makes this approach less desirable.

Frames are used as the main knowledge representation method. These are particularly appropriate for problems where we can identify a number of stereotypes, in this case relatively small number of basic map types and representation methods. Each of the blanks or 'slots' in the frame must be filled in order. This is achieved by a procedure or procedures being associated with each slot which may, for example, ask the user to answer a question, or refer to other frames (Giarranto & Riley, 1989). The resulting system has hierarchical structure where the topmost frames represent generalities and the lower ones are customised for more specific instances. Frame based systems have been identified as being particularly appropriate representations of objects in the design process (Landsdow, 1988; 1162).

Examples of the MAP frame are given in Figure 1. The generic frame (Fig. 1a) indicates the slots to be filled, the kind of information that will fill them and the procedures that are used to fill them, or that come into operation when the value in the slot is altered. A frame from the second level of the hierarchy is illustrated in Fig. 1b. There are three possible frames at this level, for 'basic', cultural and physical maps. At the third level of the hierarchy frames represent individual map types, examples being illustrated in Fig. 1c & d. Space does not allow the members of lists or details the procedures to be shown in these diagrams. The fourth and most detailed level of frames are individual map designs, which have all slots filled.

There is also a series of cartographic representation frames. Each of these contains slots for the parameters specifying the symbols in enough detail for them to be drawn. These slots are filled by default values, procedures, or by reference to look-up tables containing colour sequences, point symbols, line patterns, etc.

CONCLUSION

The main aim of the cartographic design expert system discussed here is to prevent fundamental rules of map design being broken. No attempt is made to develop a creative design tool, rather the emphasis is on imitating the design of existing maps that are effective in solving the problems associated with mapping various types of information. The formalisation of the rules of cartographic design is the critical step. Their incorporation into a frame based expert systems allows relatively similar maps to be produced with minimum effort, but the system is flexible enough to create a wide range of map types and to allow the user to interact with the design process to create the effect he wishes.

References

Bahill, A.T., Ferrell, W.R. (1986) "Teaching an Introductory Course in Expert Systems." IEEE Expert Vol.1, No.4, pp.59-63.

Begg, V. (1984) Developing Expert CAD Systems. London: Kogan Page Ltd.

Bertin, J. (1967) Semiology Graphique. Paris: Gauthier-Villars.

Ditterich, T.G., Ullman, D.G. (1987) "FORLOG: a logic based architecture for design." In Gero, J.S. (ed.) Expert Systems in Computer Aided Design. Amsterdam: Elsevier Science Publishers B.V., pp.1-24.

Forrest, D., Pearson, A.W. (1990) "Information Sources in Map Design" In Parry and Perkins (eds.) Information Sources in Cartography. London: Butterworth.

Giarranto, J., Riley, G. (1989) Expert System: Principles and Programming. Boston: PWS-Kent Publishing Company.

Keates, J.S. (1989) Cartographic Design and Production (2nd Edition). London: Longman.

Keates, J.S. (1990) "Cartographic Design and Expert Systems" Unpublished paper, Glasgow University.

Landsdow, J. (1988) "Graphics, Design and Artificial Intelligence." In: Earnshaw, R.A. (ed.) Theoretical Foundations of Computer Graphics and CAD. Berlin: Springer-Verlag.

Mackaness, W. (1986) "Detection and Heuristic Resolution of Spatial Conflicts in Digital Map Datasets." Unpublished report, Kingston Polytechnic.

Mackaness, W., Fisher, P., Wilkinson, G. (1986) "Towards a Cartographic Expert System." Proc. Auto Carto London Vol. 1, pp.578-587.

Muller, J-C., Zeshen, W. (1990) A Knowledge-Based System for Cartographic Symbol Design. Cartographic Journal Vol.27, No.1, pp.24-30.

Samson, L, Poiker, T.K. (1985) "Graphic Design with Color using a Knowledge Base." Paper presented at the 10th C.C.A. Annual Conference.

Robinson, A., Sale, R., Morrison, J., Muehrcke, P. (1984) Elements of Cartography (5th Edition). New York: Wiley.