CHORO-EXPERT

A front-end expert system determining data-appropiatness for choropleth mapping

Barend Köbben
University of Utrecht
Department of Cartography
1988

CONTENTS

1. INTRODUCTION

In recent years there has been an upsurge in the making of maps with the aid of computers. The reasons for this upsurge are the increasing ability of hard- and software to create good quality maps at decreasing costs. This development has good and bad sides.

The good side is that more maps are being made. In geography, for instance, the 'quantitative revolution' of the seventies had many geographers losing interest in maps in favour of tables and graphs. The increasing use of Geographical Information Systems has brought maps back into the picture. Programs like Atlas*Graphics(1) are popular in businesses for making maps of sales figures and the like. But most of these 'new' mapmakers have had little or no cartographic training and therefore the quality of many maps leaves much to be desired. Because you can't expect all users of these programs to take a course in basic cartography the programs themself should contain safeguards and guidelines in order to prevent the user from making basic cartographic errors. In trying to deal with this problem, two possible solutions exist (Muller, 1988).

The first is creating a cartographic tutorial program, which teaches the computer-user the basic cartographic 'grammar' and rules. This would have to be quite an extensive package, preparing the user for all cartographic taskes he could encounter in the future when using any commercial cartographic package. The second solution is making front-end systems for the cartographic programs: Interfaces between the user and the program, which prevent the user from making mistakes in the particular cartographic task the program is undertaking. The former offers a more general solution, but users of cheap and simple mapping programs can hardly be expected to spend considerable time and money on learning more than they'll probably need. The latter therefore is the more useful solution.

Such a man-machine interface should have knowledge about the cartographic rules involved in the mapping possibilities of the computer-program. It should know how to use this knowledge to make correct maps. And it should be able to handle input from the user and output to the mapping program according to these rules. These requirements make it fit the description that's usually given to an expert system (eg. Mackaness & Scott, 1987). What we need is a so-called front-end expert system

The Choro-Expert system, the subject of this paper, was made in order to test the possibilities and problems involved in making such front-end expert systems. This system carries out the limited task of checking if a selected variable of the Atlas*Graphics mapping program is appropiate for making a choropleth map. It was made using the recently released VP-Expert(2), one of the first reasonably priced expert system development tools for personal computers.

The first section of this paper is about expert systems in general and describes some of the problems involved in incorporating cartographic knowledge in a computer knowledge-base. The second section describes the Choro-Expert system and how it carries out its tasks. Finally some remarks will be made about the succes of this experiment and the lessons that can be learned from it.

2. CARTOGRAPHIC EXPERT SYSTEMS

2.1. Artificial Intelligence

Expert systems are part of the field of Artificial Intelligence, a quite recent development in computer science. Artificial Intelligence (AI) can be defined as 'the study of mental faculties through the use of computational methods' (Charniak and McDermott, 1985). One could say the ultimate goal of AI-scientist is to make a program that can 'think' and would in this respect be undistinguishable from humans. AI contains many subfields, like behaviour simulation, robotics, computer linguistics and problem solving. This last subfield is of particular interest, because of the many applications possible. Capturing the problem-solving powers of human experts in computer programs is usefull for the following reasons (Boose, 1986):

- Experts retire, taking their knowledge with them.
- If experts don't have to waste time answering questions about their present expertise, it leaves them with more time to extend their knowledge.
- Expertise would become cheaper to obtain and more widely and quicker available.
- Experts are not always consistent.

In their attempts to make problem-solving programs, scientists started with trying to build general-purpose programs and soon found only more specialised programs rendered a good problem-solving power. This shift towards specialization is shown in figure 1.

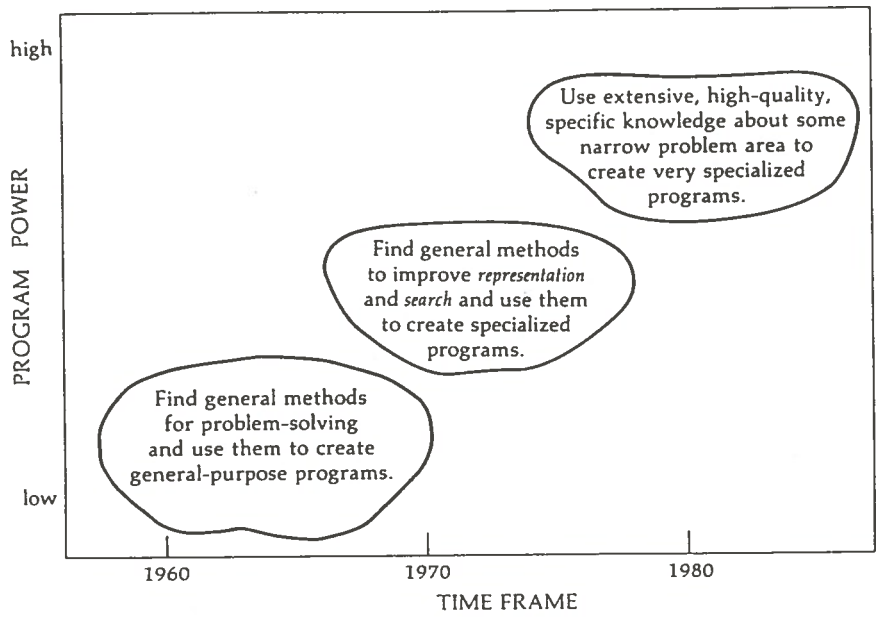

Figure 1: The shifting focus of AI (source: Waterman, 1986).

The special-purpose computer programs (upper right of figure 1), which have high-quality, specific knowledge incorporated in them, are what we call *expert systems* (Waterman, 1986).

2. 2. Expert Systems

One could say expert systems(3) are the incarnation of a human expert in a computer. The expert system can provide you with answers to questions you ask it or give you advise. Like humans, an expert system should be able to learn. This learning takes place in two forms. The first is the ability to gain new knowledge. This is something the system can't do on its own: The human expert or the user has to provide the system with this new knowledge. Secondly, the system can learn from its mistakes and successes. Expert systems basically exist of three major parts (Kraak, 1987):

- A knowledge base, in which the knowledge about the particular problems the system is supposed to solve is stored.
- A program to interpret the knowledge and relate it to information given and questions posed by the user. This is what we call the inference engine.
- A control mechanism. This provides an interface for the user and for the expert. The first one permits the user to pose questions to the systems and delivers the answers the inference engine comes up with to him. The second gives the expert the oppurtunity to provide the system with additional or changed knowledge.

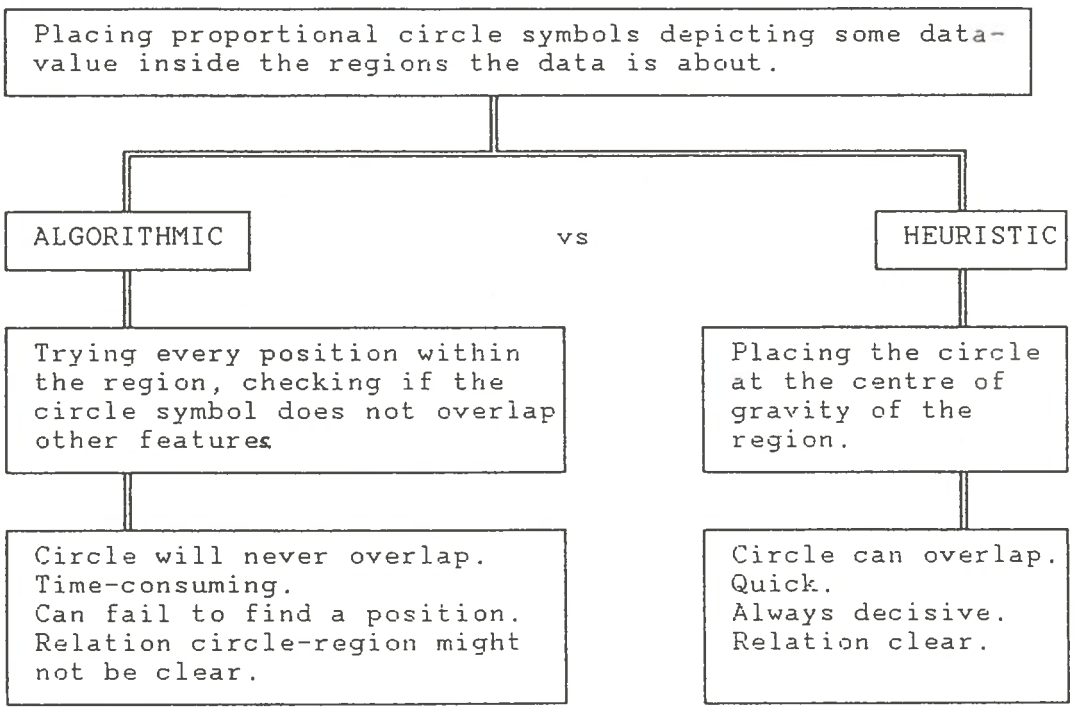

Figure 2: Algorithmic versus heuristic methods.

The knowledge base consists of two main types of knowledge: Declarative and procedural knowledge. Declarative knowledge is *knowing what*; facts and concepts that are known about the field the expert system is concerned with. These are for instance definitions, assumptions or conventions (eg. the see being blue on maps). Procedural knowledge on the other hand is *knowing how*; rules and procedures (eg. for defining class intervals or line generalisation). Some of these rules are *heuristics*. Heuristics are simplifications or rules of thumb, able to produce an acceptable solution most of the time. They are used instead of *algorithms* (formal procedures guaranteed to produce correct or optimal solutions) because in many cases these are not applicable to the specific knowledge used. The difference between algorithms and heuristics is illustrated in figure 2. For the process of name-placement on maps, for instance, an algorithm would be impractical. Therefore some heuristic rules must be used, which will most of the time place the names at convenient places, but can make a mistake every once in a while. Therefore rules and even facts in an expert system won't always be either true or false. Some uncertainty can exist, which can be incorporated in the system by using *certainty* or *confidence factors*.

The inference engine has to deal with the problem of controlling the search for solutions. It works with what we call a *search space*. This may be described as a tree, with one or more roots (the start states) spreading to a set of terminal positions or possible solutions. In most knowledge-based problem-solving there's no one correct method of reaching a solution. Therefore it would be necessary to explore several paths through the state-space and evaluating each of them. But in most situations it is impractical to explore all paths, simply because there are too many of them. So we have to apply the above-mentioned heuristic rules in order to determine which search-paths are the most likely to succeed. The search can be either *depth-first* or *breadth-first*, as shown in figure 3.

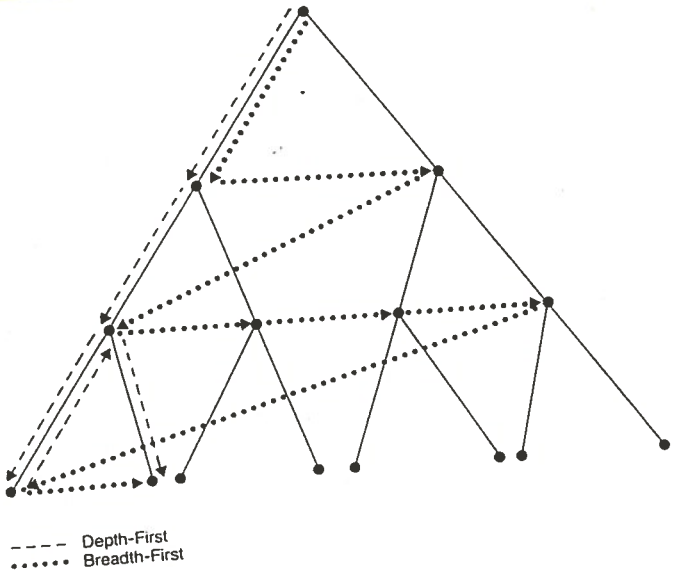

Figure 3: Depth- and breadth-fist search (source: Alty & Coombs, 1984).

Both depth-first and breadth-first searches can be conducted using *forward chaining* or *backward chaining*. Forward chaining works down from the start-state(s), trying to get to the solution. Backward-chaining on the other hand starts with the solution and then works it way up to find the start state(s). The total of start states and possible paths to the solution is called the *search space*.

Not all problems are equally well-suited to solution by expert systems. This depends on the degree in which they are 'well structured' (Alty & Coombs, 1984). Ideally structured problems would have:

- A small search space.
- Reliable domain knowledge.
- Reliable and static user-provided data.

A small search space would mean the state-space search mentioned above could be a simple exhaustive search being relatively quick and coming up with reliable solutions.

Reliable domain knowledge does not only mean the facts known have to be true, it also means it should be possible to describe it in definite, unambiguous rules. This often is one of the biggest problems of incorporating knowledge in a knowledge base.

The advantage of the data given by the user being static (ie. not changing with time) is that conclusions based upon this data will remain true throughout the consultation, making constant re-computation unnecessary.

A lot of scientific fields are not very well structured, making it difficult to capture them in expert systems. Unfortunatly, cartography is one of them, as we will see in the next paragraph.

2.3. Transfering cartographic knowledge to expert systems

Knowledge transfer to knowledge bases is one of the most important and most difficult problems involved in building expert systems.

The first step is *knowledge acquisition*. This is the process of transforming data on expertise into an *implementation formalism* (Kidd et al., 1987). In the past, most of it was done quite unsystematically and experimental, but recently more methodological approaches are being developed. Techniques like *teachback interviewing*, *verbatim protocols*, *personal construct technology* and *cognitive emulation* (for a description of these methods, see Kidd et al., 1987 and Slatter, 1987) are being used. There's even an expert system developed especially for the task of knowledge transfer (Expertise Transfer System, described in Boose, 1986).

The second step is *semantic representation*, ie. incorporating the acquired knowledge into the actual knowledge base. For this purpose the knowledge has to be represented in a format usable by the computer. The three main format types are *rules*, *frames* and *semantic nets* (these are discussed in more detail in chapter 7 of Waterman, 1986).

As Mackaness & Scott (1987) state, making maps is a multi-domained task. These domains include knowledge of geography, spatial processes, design and publication and of the perception of map users and

spatial cognition. The knowledge domains are interdependent and form a complex field of knowledge, as shown in figure 4.

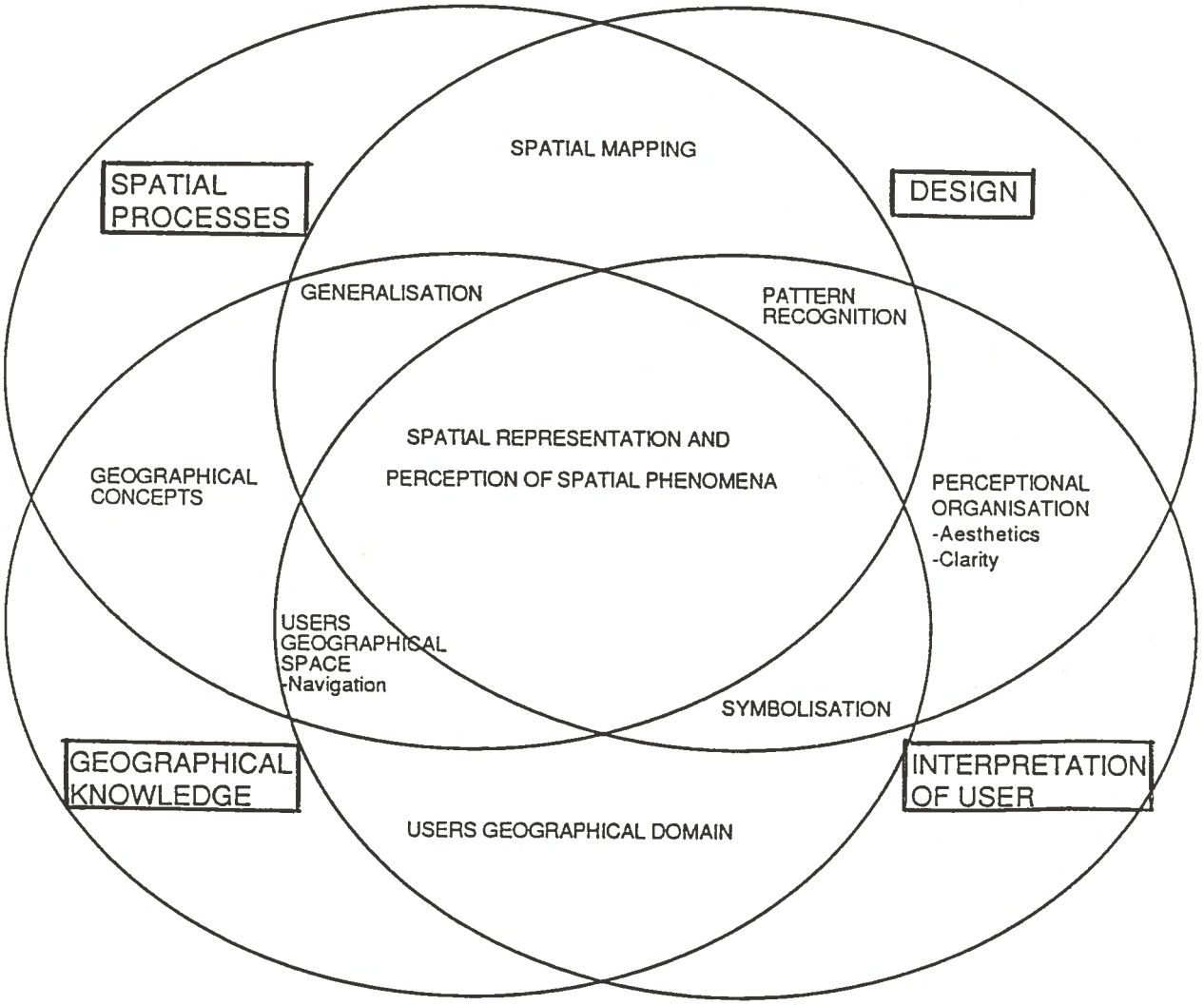

Figure 4: The interdependent nature of the sub-domains pertinent to map design (source: Mackaness & Scott, 1987).

Unfortunatly, many of these sub-domains are not well structured (see paragraph 2.2). The search space of course can be made smaller by dividing the the problem into several sub-tasks, but this leaves you with the problem of determining the order in which to execute these tasks. Often in cartography there is no such definite order.

The major problem is that there is no reliable domain knowledge. A lot of things cartographers do can't be incorporated in unamgbiguous rules. The reason for this is that much depends on perception of the map-users, spatial cognition and in particular aesthetics, which can't be semantically represented, at least not at present or in the forseeable future.

Therefore one view is, that research in cartographic expert systems should concentrate on those smaller sub-domains which can be called well structured, like automated name-placement and line-generalization. By concentrating on such smaller sub-domains, the problems remain survayable and surmountable. Cartographers who support this view (eg. Muller, 1987; 1988) state that this is the more feasible strategy, most likely to come up with useable results.

Others consider only a system that can intelligently automate the total design of maps a 'true' Cartographic Expert System (Mackaness & Scott, 1987). This holistic view of expert systems for cartography is said to represent a more general solution. But the problems involved in building such a system are gigantic and transferring cartographic knowledge of every sub-domain and semantically representing it is not only a formidable, but most likely an impossible task.

The first view is therefore, in my opinion, the most likely to lead to short- and longterm succes and is the most promising way to a solution of the problems signaled in the introduction. When the object is to prevent the users of mapping programs from making cartographic errors, the best way to reach both users and manufacturers of such programs is by providing them with practical, easy-to-use systems, aimed at their specific problems and needs. Front-end expert systems provide man-machine interfaces to do just that. The Choro-Expert system is such a system.

3. THE CHORO-EXPERT SYSTEM

3.1. Introduction

The Choro-Expert system was devised as an experiment to test the possibilities of making front-end expert systems for existing mapping programs. It checks if data chosen to map with Atlas*Graphics(4) are appropiate for choropleth mapping(5). The first thing to do is to formalise the knowledge about data-appropiatness and then semantically represent this knowledge in a format usable by the expert system. The way this has to be done is depending on the *inference engine* and the *inference method* used. In this case the inference engine and method of the VP-Expert program.

3.2. VP-Expert

The rule-based expert system development tool VP-expert was introduced by Paperback Software in 1987 and was one of the first to offer an affordable, easy-to-use expert system development tool for PC's (PC Business Info, 1987).

You build your expert system by making a file (with extension .KBS), containing the knowledge base and various statements allowing you to control in- and output, questioning and the like. When running the expert system, VP-Expert interprets this file. Therefore the total VP-Expert system always has to be present when consulting the system (some other development tools compile, thus needing only a small runtime-program present at the time of consultation).

The semantic representation of knowledge is in the format of If-Then-Else rules. The inference engine uses backward-chaining, although the manual also mentions a not very practical method of working with forward-chaining (VP-Expert manual, 1987).

The communication with the user and the execution of the search for solutions is contained within the so-called *Actions-block*. The system is very 'open', ie. it's possible to call external programs(6) and communicate with data- and textfiles of popular programs(7). This means, that developing an expert system with VP-Expert will usually involve some programming in a computer-language like Pascal, Fortran or compiler-Basic.

The system offers many options, like the possibility to induce knowledge-bases from tables and data-bases and tracking the search-path of the inference engine. Particulary usefull is the possibility to provide If-Then-Else rules with a 'because' clause. If the user, when posed a certain question, asks why, he is shown this 'because' clause. All in all it's quite a powerful package concerning the relatively low price (about $149).

Other available expert system development tools are more expensive (ranging form $495 to $5000 or more) and sometimes require AT-compatibles or extended memory. For a sampling of available systems, see Bender, 1987 and Waterman, 1986.

Knowing the possibilities of the system, the next step is to formalise the knowledge about data-appropiatness for choropleth maps.

3. 3. Data appropiateness for choropleth maps

Choropleth mapping (also called *area* or *shaded* mapping) is a widespread and popular form of mapping quantative data. It is described by the International Cartographic Association as 'a method of cartographic representation which employs distinctive colour or shading applied to areas other than those bounded by isolines. These are usually statistical or administrative areas'. Choropleths are easy to use and very flexible, showing the reader information about the value classes associated with a geographical area, or the actual values in the case of unclassed choropleths (Tobler, 1973), giving him a sense of the overall geographical pattern of the mapped variable and giving him the possibility to compare two choropleth maps (eg. of different years) with each other. But it is important that the data used for making choropleths are appropiate, otherwise the reader is provided with misleading information.

In this experiment, the method of knowledge acquisition was straightforward. The knowledge about data-appropiatness is well-developed and formalised. Therefore the decision was made to extract it from existing literature about data-appropiatness and its problems (eg. Campbell, 1984; Cuff & Mattson, 1982; Dent, 1985; Thauer, 1980; Tobler, 1973). From the literature the conclusion was drawn that data, in order to be appropiate should be:

- occuring in or being attributable to definite enumeration units, like administrative units (states, countries) or statistical units (census tracts, dutch COROP's).
- derived values (ratios), as opposed to total values (rates).
- having values that are uniformly spread throughout the mapping unit.

The first requirement is quite straightforward. The distribution of values of phenomena that are continuous in nature, like rainfall, are not controlled by the enumeration units. It would therefore be nonsense to map them using these units.

The second is important, because most of the time the areas involved are not equal in size. When mapping total values, uniform distributions may be masked or non-existing differences may be suggested, as shown in figure 5.

The third requirement is particulary important, because the choropleth mapping technique is insensitive to changes of the value within the area. Therefore the actual values of the variable found throughout the area should not deviate too much from the value chosen to represent this area.

The first two requirements can quite easily be semantically represented in a knowledge base. This is not the case with the third one. Mapping the population density of the countries of the European Community, for instance, is allright. But doing the same with Northern America is not. Giving one density value for Canada is nonsense, giving the enormous differences between the quite densely populated south and the vast emptiness of the Northern Territories. But there is no clear boundary between the deviation being insignificant and being too big.

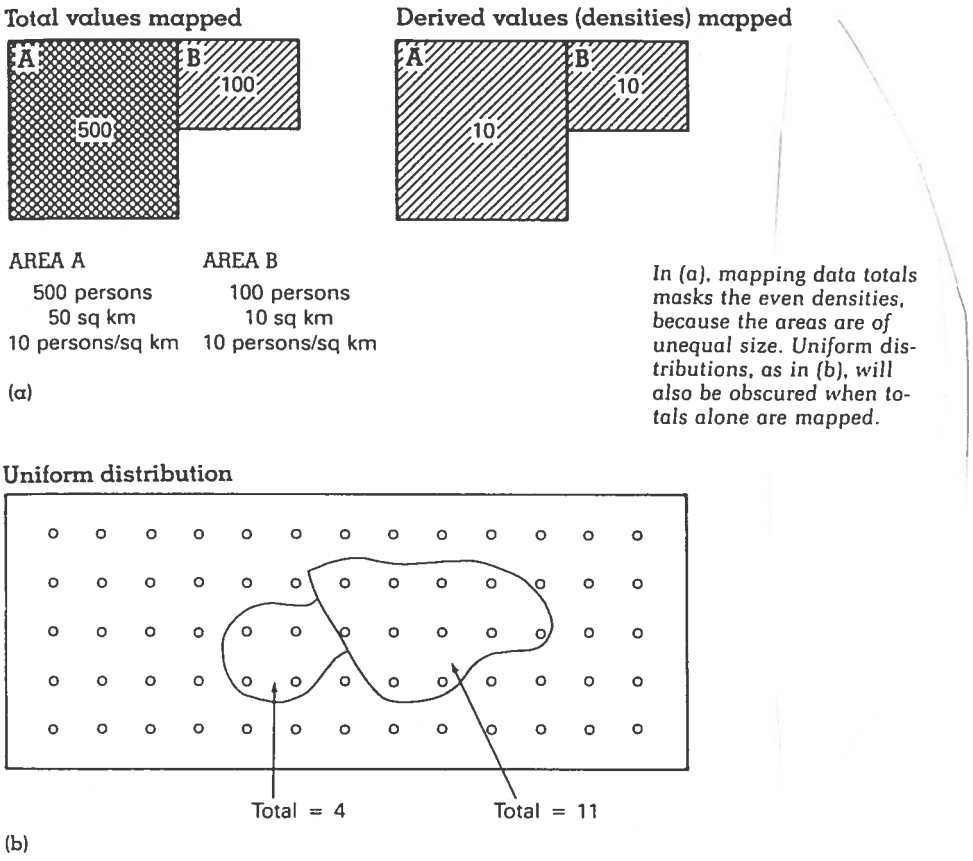

Figure 5: Reasons for using only derived values (source: Dent, 1987).

Thauer recognised this and other problems (eg. the changing of boundaries with time) involved with choropleth mapping and suggested the *dasymetric* map as a possible solution (Thauer, 1980).

3.4. Implementation

In this paragraph the way in which the Choro-Expert system actually executes its task will be described in the same sequential order in which the system works. A simplified flow-chart of the consultation process can be found in figure 6.

First a batch-file is called (SHOWDIR.BAT), which shows all datafiles available. Then the program MAKEFILE.COM(8) is called. This lets the user choose a datafile and a particular variable from that file. It then puts the relevant information about the variable (such as name, values, the regionnames connected to it) in a file (FACTFILE) contained in VP-Expert variables. This is relatively simple, because Atlas*Graphics files as well as the factfile are in ASCII-format. Subsequently this information is loaded into the knowledge base.

Now the system asks the user to describe the kind of variable he wants to map. It then checks in its database (CHORO.DBF) if it has encountered this one before.

This database(9) contains three fields: Datatype, OK1 and OK2. The first is the description of the kind of variable (eg. population density, birth rate). The second and third respectively state if this kind of variable has in the past been found to be appropiate

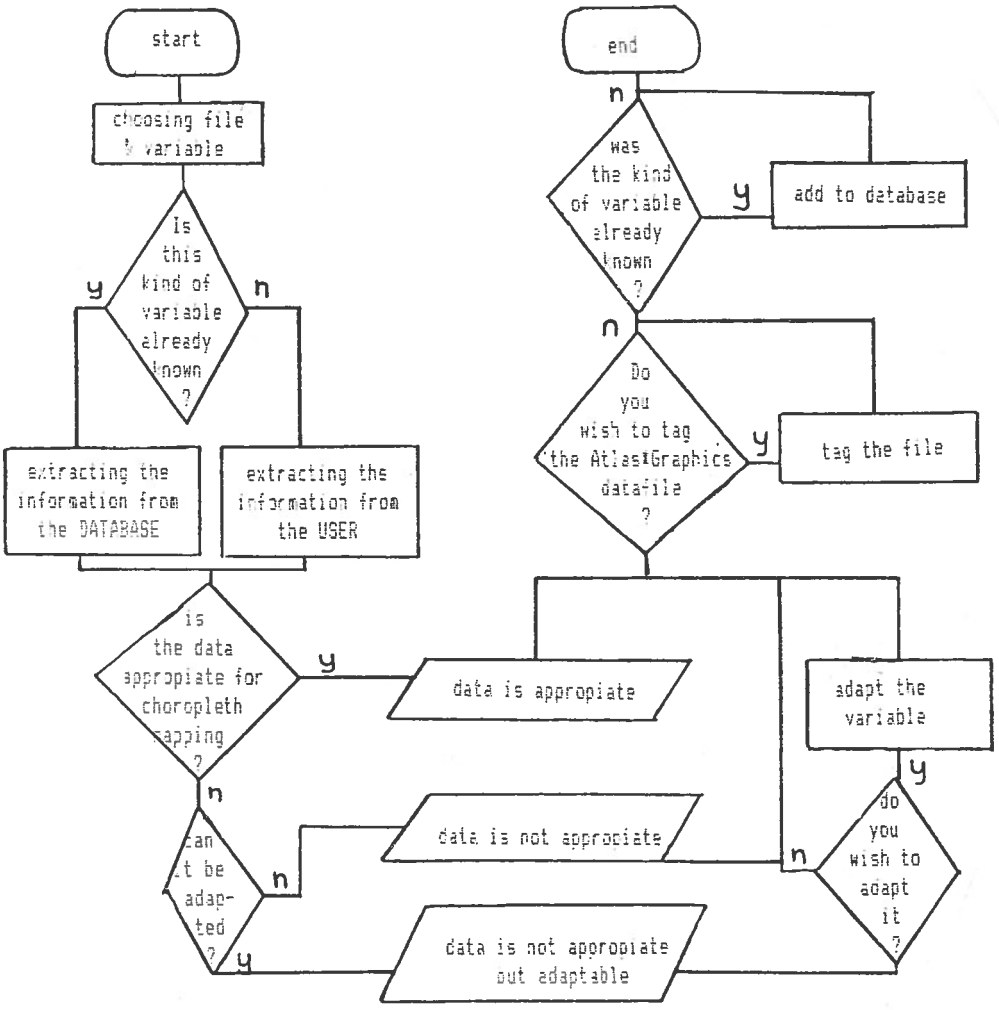

Figure 6: Simplified flow chart of a Choro-Expert consultation.

for choropleth mapping and if not, if it can be adapted. If it has encountered the datatype before, it looks in the database if the data is appropiate for choropleth mapping and provides the user with the answer. If it hasn't, it will ask the user several questions about the variable and then decides on the data-appropiatness. The results are stored in the database. In this way the system actually increases its knowledge with every consultation and therefore can be said to be able to learn.
In testing for data-appropiatness the criteria mentioned in paragraph 3.3 are used, with an additional check on the data being *quantitative* (bearing in mind choropleth mapping is a technique usable for mapping quantitative data only). In order to find out if the data are occuring in or attributable to definite enumeration units

the system ask for the nature of the *features* (the Atlas*Graphics name for polygons). It offers twelve possibilities, some specific (eg. county), some more general (eg. political unit). Of course this can' t be exhaustive, but seems quite satisfactory. The procedure is the same with the question about the data being derived values. Finding out if the data is uniformly spread turned out to be difficult. The chosen solution(10) is arbitrary and not very satisfactory. Better results may be possible if the difference in area of the regions would in some way be taken into account, by accessing the boundary files of Atlas*Graphics. These are in a special binary format. Converting them to ASCII-files and subsequently by some algorithm finding out the respective areas is possible, but not directly from the Choro-Expert system. The user would have to exit Choro-Expert, enter the Boundary-Edit portion of Atlas*Graphics in order to convert the boundary-files and then re-enter the Choro-expert system. It was decided this would conflict with the idea of a transparant interface and was therefore not implemented.

After having obtained the answers the system decides if the data is either appropiate or not appropiate for choropleth mappping. All criteria have to be met, if one fails the data is labeled unappropiate. Sometimes the data may be not appropiate for choropleth mapping, but could be adapted to be so. This happens when all criteria but the ' derived values' -one are met. In some cases the totals can easily be changed to derived values (eg. dividing total population by land area, giving you population density). Choro-Expert either can perform this operation itself (together with ADAPT.COM) or refer the user to the Data-Edit portion of Atlas*Graphics(11).

Finally, the variable in the Atlas*Graphics datafile is accordingly tagged (with a ' +' in case of appropiateness, with a ' -' if not appropiate) by calling the program UPDATE.COM. This way, the user of Atlas*Graphics knows when a variable he wants to select for mapping has been ' approved' by the Choro-Expert system or not. Ideally, Atlas*Graphics would refuse to load the variables tagged as being not appropiate. But for this to be possible the mapping program itself should be altered, requiring acces to the source-code and coöperation from the software company.

4. CONCLUSIONS

Developing the Choro-Expert system made several things clear. The first is, that building these specialized, front-end expert systems is very well possible, though it takes more time and effort than was expected. A very important step in the process is the defining of the task and the formalising of the knowledge about this task. In this specific experiment this was not too much of a problem, because the knowledge about data-appropiatness is well-developed and formalised in cartographic literature.

The results are quite positive, though some aspects of the Choro-Expert system could be improved. The user should have more choices when asked the questions about data-appropiateness, thus requiring less knowledge of for instance statistics(12). This could be reached by letting the user give a specific description of for instance the nature of the data-values and then trying to locate this in a database (a procedure similar to the one used for finding out the kind of variable). Incorporating the areas of the regions, as mentioned before, could also improve the performance of the system.

A second conclusion is, that although the VP-Expert system is quite useful for familiarizing oneself with the concepts and implementation of expert systems, it is not powerful enough to handle more complex problems. Controlling the program flow is difficult. There's one simple loop-procedure (the Whileknown-End statement), forcing the programmer to introduce various superfluous variables when he wants some repetitive action, thus making the program-structure untransparent. Furthermore, the I/O-handling is very basic. When developing an expert system for a more demanding task than the one described in this paper, one would need a system with more possibilities, especially concerning program-flow control and I/O-handling.

Finally, one must realise that with appropiate data the user of Atlas*Graphics is not guaranteed to make good maps. Other decisions involved in making choropleth maps, such as classification, choice of shading and color, require cartographic guidance. The importance of providing commercial mapping programs with mechanisms to prevent the user from making cartographic mistakes cant't be overestimated. Ideally, the writers of these programs should incorporate this in their programs from the start. More coöperation between cartographers and software developers therefore is very important.

NOTES

1. Atlas*Graphics is a trademark of STSC, Inc.
2. VP-Expert is a trademark of Paperback Software International
3. Recently, there is a tendency to replace the term 'expert system' with 'knowledge system'. In this paper, however, 'expert system' is used, mainly to prevent confusion with 'knowledge base', which is a part of an expert system.
4. Since AtlasAmp datafiles are structured exactly like Atlas*Graphics datafiles, the system works with these as well.
5. Atlas*Graphics can produce two types of maps: Choropleth maps and dot maps. But the possibilities of making dot maps are very limited and not very well implemented. Therefore they were not part of this experiment.
6. Program calls can be made to programs with .COM, .EXE and .BAT extensions. However, I discovered a bug in the calling of DOS batch-files (.BAT). VP-expert seems to be able to call only batch files on harddisk, ignoring calls to drives a: and b:.
7. VP-Expert can read and write files from DBase and clones, Lotus 123 and clones and ASCII-textfiles.
8. This program and the programs update.com and adapt.com were written in Turbo-Pascal (trademark of Borland International).
9. This is a database in DBase III-format, though it was originally created using RapidFile (both are trademarks of Ashton Tate). Communication with the database is possible through the commands GET, PUT and APPEND.
10. The question being asked is: 'Would you say the data-value chosen to represent each region is representative for the whole of that region, or would you find deviation from it within the region ?'
    The choices offered are: Representative, some deviation and very big deviation.
11. When using some datafiles, the system tends to crash or gives an out of memory error. The size of the files can hardly account for that. Possibly this is a bug in VP-Expert.
12. In the present situation, the user has to be familiar with terms like 'derived values' and 'ratios'.

GLOSSARY

Most definitions are derived from Waterman (1986), expect when stated different.

Algorithm - Formal procedure, guaranteed always to produce a correct or optimal solution to a problem.
Artificial Intelligence - The study of mental faculties through the use of computational methods (Charniak & McDermott, 1985).
Backward chaining - An inference method where the system starts with what it wants to prove (eg. Z) and tries to establish the facts it needs to prove Z. These are typically given in rule form (eg. if A and B then Z). If A and B aren't known, the system will try to prove A and B by establishing any additional facts (as specified by other rules) needed to prove them. The additional facts are established the same way A and B were established and the process continues until all facts are established or the system gives up in defeat.
Certainty factor - A number that measures the certainty or confidence one has that a fact or rule is valid.
Choropleth - A method of cartographic representation which employs distinctive colour or shading applied to areas other than those bounded by isolines. These are usually statistical or administrative areas (International Cartographic Association).
Confidence factor - see: certainty factor.
Declarative knowledge - Facts and concepts known about the specific field the system is concerned with: 'knowing what' (Alty & Coombs, 1984).
Domain knowledge - Knowledge about the problem domain.
Expert system - Program solving programs capable of representing and reasoning about some knowledge rich domain, with a view to solving problems and giving advice (Mackaness & Scott, 1987).
Forward chaining - An inference method where the IF-portions of rules are matched against facts to establish new facts.
Frames - A semantic representation method that associates features with nodes representing concepts or objects. The features are described in terms of attributes (called slots) and their values. The nodes form a network connected by relations and organized into a hierarchy. Each node's slots can be filled with values to help describe the concept that the node represents. The process of adding or removing values from the slots can activate procedures attached to the slots. These procedures may then modify values in other slots, continuing the process until the desired goal is achieved.
Heuristic - Simplification or rule of thumb, which produces an acceptable solution to a problem most of the time.
Inference chain - The sequence of steps or rule applications used by a rule-based system to reach a conclusion.
Inference engine - That part of an expert system that contains the general problem-solving knowledge. The inference engine processes the domain knowledge to reach new conclusions.
Inference method - The technique used by the inference engine to access and apply the domain knowledge, eg. Forward- or Backward chaining.
Knowledge base - The portion of an expert system that contains the domain knowledge.
Procedural knowledge - Rules and procedures necessary to work with declaritive knowledge: 'knowing how' (Alty & Coombs, 1984).

Rule - A semantic representation method that formally specifies a recommandation, directive or strategy, expressed as IF premise THEN conclusion 1 (ELSE conclusion 2), or IF condition THEN action 1 (ELSE action 2).
Search space - The set of all possible inference chains.
Semantic Net - A semantic representation method consisting of a network of nodes, standing for concepts or objects, connected by arcs describing the relationships between the nodes.
Semantic representation - Incorporating the domain knowledge in the knowledge base in a computer-usable format (Boose, 1986).
State - A step in the inference chain (Alty & Coombs, 1984).

REFERENCES

Alty, J. L. & M. J. Coombs.
Expert systems, concepts and examples.
NCC Publications (1984).

Bender, Eric.
The knowledge engineers.
In: PC World, september (1987).

Boose, John H.
Expertise transfer for expert system design.
Elsevier (1986).

Campbell, John.
Introductory Cartography.
Prentice-Hall (1984).

Charniak, E. & D. V. McDermott.
Introduction to artificial intelligence.
Addison-Wesley (1985).

Cuff, David J. & Mark T. Mattson.
Thematic maps: Their design and production.
Methuen (1982).

Dent, Borden D.
Principles of thematic map design.
Addison-Wesley (1985).

Kidd, Alison L. et al.
Knowledge acquisition for expert systems; A practical handbook.
Plenum Press (1987).

Kraak. M. J.
Kunstmatige intelligentie, kennissystemen en kartografie.
In: Proceedings studiedag NVK-werkgroep computerkartografie (1987).

Mackaness, M. A. & D. J. Scott.
The problems of operationally defining the map design process for cartographic expert systems.
In: Proceedings studiedag NVK-werkgroep computerkartografie (1987).

Muller, J-C.
Applications of knowledge based systems to cartography: prospects and limitations.
In: Proceedings studiedag NVK-werkgroep computerkartografie (1987).

Muller, J-C.
Proposal for the development of a computer based tutorial cartographic system.
Internal report at ITC Enschede (1988).

Slatter, P. E.
Building experts systems: Cognitive emulation.
Halsted press (1987).

Thauer, Walter.
Atlasredaktion in Zusammenspiel von Kartographie, Geographie und Regionalstatistik.
In: Internationales Jahresbuch für Kartographie, vol. 20 (1980).

Tobler, Waldo R.
Choropleth maps without class intervals?
In: Geographical Analysis, V, no. 3 july (1973).

Waterman, Donald A.
A guide to expert systems.
Addison-Wesley (1986).

-
VP Expert: Allemaal aan het expert systeem.
In: PC Business Info 2/87 (1987).

-
VP-Expert Manual
Paperback Software International (1987).

MANUAL OF CHORO-EXPERT

Consulting the Choro-Expert system is very straightforward. It runs on any PC, AT or compatible with a harddisk and 640 Kbyte of memory, using DOS 3.2. The files needed to run the system are:

- All VP-Expert system-files (VPXHELP-files can be ommited if not needed).
- CHORO.KBS: This is the actual knowledge base, including the program-flow control.
- CHORO.DBF: The database with the variables encountered in previous consultations. This database is expanded every session. If necessary, changes or additions can be made to it, using DBase II, III, IIIplus or VP-INFO.
- CHORO.BAT, SHOWDIR.BAT, MAKEFILE.COM, UPDATE.COM, ADAPT.COM and NOFACTS.VPX: These are the supplementary programs, handling I/O and the like.
- INSTALL.COM: To install Choro-Expert.

Copy the files to your harddisk. Use INSTALL to install the system. To start a consultation type 'choro'. Choose an Atlas*Graphics datafile and a variable from that file. The file is then loaded into the knowledge base. Now the system asks for a description of the kind of variable. To help you, the Atlas*Graphics name of this variable is shown at the top. Note that the description must be 20 characters or shorter.

After this either the system either gives you its conclusions on the appropiatness now (when it's encountered this variable before) or asks you several questions about it. Questions can be answered by moving the highlight with the cursor-keys to the desired answers and pressing ENTER. When you want, you can provide your answer with a *confidence factor* between 0 and 100 by pressing HOME first, entering the factor and then pressing ENTER. When you want to know why a question is asked, press /W. A short explanation is then given.

After the system has told you if the data is appropiate you can quit the consultation with or without tagging the variable in the Atlas*Graphics file. A third option is provided when the variable is not appropiate, but could possibly be adapted. In order to do this, select 'Adapt the variable' and answer the questions following.

Finally, the system updates the database if necessary and the consultation ends.

For questions concerning Choro-Expert you can contact:

Barend Köbben
C. Roobolstraat 90
3554 BW Utrecht
030-445199

PROGRAM SOURCES

CHORO.KBS (the file interpreted by VP-Expert during consultation).

```
RUNTIME;
EXECUTE;
ENDOFF;
ACTIONS
     COLOR = 0
     DISPLAY"------------------------------------------------------------

                                CHORO-EXPERT version 1.4
                                  (c) B.J.Kobben 1988

This expert-system helps the user of the ATLAS*GRAPHICS (trademark
of STSC, Inc) mapping program to determine if the data to be mapped
is appropiate for a choropleth map.
It works in conjunction with the programs MAKEFILE.COM, UPDATE.COM,
ADAPT.COM, SHOWDIR.BAT and the files NOFACTS.VPX and CHORO.DBF.

------------------------------------------------------------------"
     DISPLAY""
     DISPLAY""
     DISPLAY""
     DISPLAY""
     DISPLAY"                                  Press any key to continue...~"
     WHILEKNOWN no_factfile_read
          BCALL showdir
          DISPLAY "Loading Datafile..."
          LOADFACTS factfile
          RECEIVE longname, chosen_var_name
          RECEIVE longname, feature_description
          FIND factfile_loaded?
          RESET factfile_loaded?
          END
      CLS
      rec = 1
      recs_displayed = 0
      COLOR = 15
      DISPLAY"{chosen_var_name}"
      COLOR = norm_color
      DISPLAY""
      FIND variable_description
      WHILEKNOWN datatype
          GET variable_description = datatype,choro,ALL
          FIND match
          END
      FIND already_known
      DISPLAY"=========================================================-
======================"
      DISPLAY""
      FIND conclusion
      DISPLAY""
      DISPLAY"=========================================================-
======================"
      FIND action
      DISPLAY""
      DISPLAY"KARTO-EXPERT is done."
      DISPLAY"Press any key to return to system...~"
;
{=======================================================
RULE match?
IF   datatype <> UNKNOWN
THEN match = yes
     choropleth_OK = (OK1)
     adapting = (OK2)
ELSE match = no
;

RULE already_known?
IF
     match = yes
THEN
     already_known = yes
```

```
ELSE
     already_known = no
     CLS
     COLOR =15
     DISPLAY"{feature_description}"
     COLOR = norm_color
     WHILEKNOWN not_all_regions
          FIND last_region?
          RESET last_region?
          END
     DISPLAY"{recs_displayed} out of {number_of_recs} shown."
     DISPLAY"-------------------------------------------------------
----------------------"
     FIND choropleth_OK
     datatype = (variable_description)
     OK1 = (choropleth_OK)
     OK2 = (adapting)
     APPEND choro
;
! -------- RULES FOR LOOPS -----------------------------
! ------------------------------------------------------
RULE fileread?
IF   file_read = no
THEN factfile_loaded? = no
     no_factfile_read = yes
     DISPLAY"File can't be opened!"
     DISPLAY""
     DISPLAY"Press /Q to end "
     DISPLAY"or any other key to retry...~"
ELSE factfile_loaded? = yes
     RESET no_factfile_read
;
! ------------------------------------------------------
RULE show_regions
IF
     rec < 14 AND rec <= (number_of_recs)
THEN
     not_all_regions = yes
     last_region? = no
     regio1 = (regionname[rec])
     rec1 = (rec+13)
     FIND second_row_regio
     RESET second_row_regio
     DISPLAY "{20regio1}{40regio2}"
     rec = (rec+1)
ELSE
     last_region? = yes
     RESET not_all_regions
;
! ------------------------------------------------------
RULE second_row_regio?
IF   regionname[rec1] <> UNKNOWN
THEN regio2 = (regionname[rec1])
     recs_displayed = (recs_displayed + 2)
     second_row_regio = yes
ELSE regio2 =
     recs_displayed = (recs_displayed + 1)
     second_row_regio = no
;
! ------------------------------------------------------
RULE show_variables
IF
     variablenr <= (number_of_variables)
THEN
     not_all_variables = yes
     last_variable? = no
     RESET varname
     RECEIVE longname, varname
     FIND variable_display
     RESET variable_display
     variablenr = (variablenr+1)
ELSE
     last_variable? = yes
     RESET not_all_variables
     variablenr = (variablenr-1)
;
```

```
! ------------------------------------------------------
RULE displayhow?
IF   variablenr = (workvariable)
THEN COLOR = 15
     DISPLAY"{varname}"
     variable_display = highlighted
     COLOR = norm_color
ELSE DISPLAY"{varname}"
     variable_display = normal
;
! ------------------------------------------------------
RULE show_data
IF
     rec < 12 AND rec <= (number_of_recs)
THEN
     not_all_data = yes
     last_data? = no
     rec1 = (rec + 11)
     FIND second_row_data
     RESET second_row_data
rec = (rec+1)


ELSE
     last_data? = yes
     RESET not_all_data
;
! ------------------------------------------------------
RULE second_row_data1
IF   variabledata[rec1] <> UNKNOWN
THEN DISPLAY"{20regionname[rec]}:{16variabledata[rec]}{22regionname-
[rec1]}:{16variabledata[rec1]}"
     recs_displayed = (recs_displayed + 2)
     second_row_data = yes
ELSE DISPLAY"{20regionname[rec]}:{16variabledata[rec]}"
     recs_displayed = (recs_displayed + 1)
     second_row_data = no
;
! ------------------------------------------------------
! ------ DATA APPROPIATENESS TESTING --------------------
! ------------------------------------------------------
RULE 4a
IF   passed_3 = yes AND
     spread = representative OR
     spread = some_deviation
THEN choropleth_OK = yes
BECAUSE "Changes of the value of the variable within the features
cannot be detected on a choropleth map. Therefore one shouldn't
make a choropleth map if these differences are to big."
;
! ------------------------------------------------------
RULE 4b
IF   passed_3 = yes AND
     spread = very_big_deviation OR
     spread = not_known
THEN choropleth_OK = no
     adapting = not_possible
;
! ------------------------------------------------------
RULE 4c
IF   passed_3 = no AND
     spread = not_known OR
     spread = very_big_deviation
THEN choropleth_OK = no
     adapting = not_possible
;
! ------------------------------------------------------
RULE 4d
IF   passed_3 = no AND
     spread = representative OR spread = some_deviation
THEN choropleth_OK = no
;
! ------------------------------------------------------
RULE 3a
IF   passed_2 = yes AND
     data_nature = ratios OR
```

```
     data_nature = densities OR
     data_nature = proportions OR
     data_nature = percentages OR
     data_nature = derived_values OR
     data_nature = averages
THEN passed_3 = yes
     adapting = unnecessary
BECAUSE "Choropleth maps should be made only when the data to be
mapped are derived values (ratios). The choropleth mapping of totals
is unacceptable, because the difference in size of the enumeration
units
would alter the impression of distribution."
;
{-------------------------------------------------------
RULE 3b
IF   passed_2 = yes AND
     data_nature = totals OR
     data_nature = not_known
THEN passed_3 = no
     adapting = possible
ELSE passed_3 = no
     choropleth_OK = no
     adapting = not_possible
;
{-------------------------------------------------------
RULE 2a
IF   passed_1 = yes AND data_sort = quantitative
THEN passed_2 = yes
BECAUSE "Choropleth maps should be made only when the data is quan-
titative.  You Gshouldn't make a choropleth with qualitative data."
;
{-------------------------------------------------------
RULE 2b
IF   passed_1 = yes AND
     data_sort = qualitative OR
     data_sort = not_known
THEN passed_2 = no
;
{-------------------------------------------------------
RULE 1a
IF   geocomponent = administrative_unit OR
     geocomponent = political_unit OR
     geocomponent = statistical_unit OR
     geocomponent = country OR
     geocomponent = state OR
     geocomponent = province OR
     geocomponent = county OR
     geocomponent = region OR
     geocomponent = municipality
THEN passed_1 = yes
     CLS
     COLOR = 15
     DISPLAY"{chosen_var_name}"
     COLOR = norm_color
     rec = 1
     recs_displayed = 0
     WHILEKNOWN not_all_data
          FIND last_data?
          RESET last_data?
          END
     DISPLAY"{recs_displayed} out of {number_of_recs} shown."
     DISPLAY"-------------------------------------------------------
----------------------"
BECAUSE "Choropleth maps should be made only when discrete data
occur in or can Gbe attributed to definite enumeration units (eg.
number of pensioners per country). GYou can't map data that are
continuous in nature (eg. average temperature), because their dis-
tribution is not Gcontrolled by administrative or statistical subdi-
visions."
;
{-------------------------------------------------------
RULE 1b
IF   geocomponent = not_known
THEN passed_1 = yes
     geocomponent = feature
     CLS
```

```
     DISPLAY"{chosen_var_name}"
     rec = 1
     recs_displayed = 0
     WHILEKNOWN not_all_data
          FIND last_data?
          RESET last_data?
          END
     DISPLAY"{recs_displayed} out of {number_of_recs} shown."
     DISPLAY"-------------------------------------------------------
----------------------"
;
!-----------------------------------------------------
RULE 1c
IF   geocomponent = measurement_point OR
     geocomponent = datacollection_point
THEN passed_1 = no
;
!-----------------------------------------------------
!----------  CONCLUSION BLOCK  -----------------------
!-----------------------------------------------------
RULE conclusion1
IF   choropleth_OK = yes
THEN DISPLAY "The variable you selected is appropiate for choropleth
mapping."
     conclusion = given
;
!-----------------------------------------------------
RULE conclusion2
IF   choropleth_OK = no AND
     adapting = possible
THEN DISPLAY "The variable you selected is NOT appropiate for cho-
ropleth Gmapping. However, it may be possible to adapt it."
     conclusion = given
;
!-----------------------------------------------------
RULE conclusion3
IF   choropleth_OK = no AND
     adapting = not_possible
THEN DISPLAY "The variable you selected is NOT appropiate for cho-
ropleth Gmapping."
     conclusion = given
;
!-----------------------------------------------------
!---- ACTION DECISION BLOCK ------------------------
!-----------------------------------------------------
RULE action1
IF   ask_action = Adapt_the_variable AND adapting = possible
THEN action = adapt
CLS
DISPLAY"Variables in {datadir}\{AG_data-file} (highlighted = chosen
variable)"
variablenr =1
WHILEKNOWN not_all_variables
          FIND last_variable?
          RESET last_variable?
          END
DISPLAY"-------------------------------------------------------------
-------------------"
DISPLAY"The chosen variable is in the form of {data_nature}"
FIND adaption
;
!-----------------------------------------------------
RULE action2
IF   ask_action = Adapt_the_variable AND adapting = not_possible
THEN DISPLAY"Adapting not possible..."
     action = none
;
!-----------------------------------------------------
RULE action3
IF   ask_action = Adapt_the_variable AND adapting = unnecessary
THEN DISPLAY"Adapting not necessary..."
     action = none
;
!-----------------------------------------------------
RULE action4
IF   ask_action = File_result_&_quit
```

```
THEN action = file
     SHIP tempfact,datadir
     SHIP tempfact,AG_data-file
     SHIP tempfact,number_of_variables
     SHIP tempfact,workvariable
     SHIP tempfact,choropleth_OK
     SHIP tempfact,adapting
     CCALL update
;
!-----------------------------------------------------
RULE action5
IF   ask_action = Quit
THEN action = quit
;
!-----------------------------------------------------
RULE adaptioncalling1
IF   density_possible = yes
THEN adaption = done
     FIND divide_variable
     SHIP tempfact,datadir
     SHIP tempfact,AG_data-file
     SHIP tempfact,number_of_variables
     SHIP tempfact,workvariable
     SHIP tempfact,choropleth_OK
     SHIP tempfact,adapting
     CCALL update
     SHIP tempfact,AG_data-file
     SHIP tempfact,number_of_variables
     SHIP tempfact,workvariable
     SHIP tempfact,divide_variable
     CCALL adapt
     choropleth_OK = yes
     adapting = unnecessary
     DISPLAY"A new variable has been created and tagged as being
appropiate for a choropleth map."
     DISPLAY"The original variable has been tagged as being  NOT
appropiate for a choropleth map."
;
!-----------------------------------------------------
RULE adaptioncalling2
IF   density_possible = no OR density_possible = not_known
     AND ratio_possible = yes
THEN SHIP tempfact,datadir
     SHIP tempfact,AG_data-file
     SHIP tempfact,number_of_variables
     SHIP tempfact,workvariable
     SHIP tempfact,choropleth_OK
     SHIP tempfact,adapting
     CCALL update
     DISPLAY"=====================================================-
======================="
     DISPLAY""
     DISPLAY"Enter the DATA-EDIT portion of ATLAS*GRAPHICS and
perform this mathematical operation."
     DISPLAY""
     DISPLAY"=====================================================-
======================="
     adaption = suggested
ELSE SHIP tempfact,datadir
     SHIP tempfact,AG_data-file
     SHIP tempfact,number_of_variables
     SHIP tempfact,workvariable
     SHIP tempfact,choropleth_OK
     SHIP tempfact,adapting
     CCALL update
     DISPLAY"Adapting is NOT possible."
     DISPLAY""
     adapting = not_possible
     adaption = not_done
     choropleth_OK = no
;
!-----------------------------------------------------
!---------------- QUESTIONING ------------------------
!-----------------------------------------------------
ASK geocomponent:"What is the nature of these features ?";
CHOICES geocomponent: country, state, province, region, county,
```

```
municipality,administrative_unit, political_unit, statistical_unit,
measurement_point, datacollection_point, not_known;

ASK spread:
"Would you say the data-value chosen to represent each {geocompo-
nent}
is representative for the whole of the {geocomponent},
or would you find deviation from it within the {geocomponent} ?";
CHOICES spread: representative, some_deviation, very_big_deviation,
not_known;

ASK data_sort: "Is the data quantitative or qualitative ?";
CHOICES data_sort: quantitative, qualitative, not_known;

ASK data_nature:
"What is the nature of the data ?";
CHOICES data_nature: ratios, densities, proportions, percentages,
averages, derived_values, totals, not_known;

ASK ask_action: "What do you want to do now:";
CHOICES ask_action: file_result_&_quit, quit, adapt_the_variable;

ASK ask_action_again: "What do you want to do now:";
CHOICES ask_action_again: file_result_&_quit, quit;

ASK density_possible:
"Could you change this to any form of ratio (eg. density) by divi-
ding it by one of the other variables (eg. area) ?";

ASK ratio_possible:
"Could you change it to a ratio by another mathematical operation
with any other variable ?";

CHOICES enumeration_unit, density_possible, ratio_possible:
yes, no, not_known;

ASK divide_variable: "Which variable do you want to divide it by
(1-{number_of_variables}) ?";

ASK variable_description: "Give a suitable description of this kind
of variable (max. 20 characters):";
```

Choro.bat (to start a consultation).

```
echo off
vpx choro
if exist factfile del factfile
if exist longname del longname
if exist tempfact del tempfact
```

Showdir.bat (To show a directory of ATLAS*GRAPHICS data-files and start makefile.com).

```
echo off
rem c:\vpexpert
rem c:\atlas\data
copy nofacts.vpx factfile.
if exist longname. del longname.
if exist tempfact. del tempfact.
cls
dir/w c:\atlas\data\*.dat
c:\vpexpert\makefile
echo on
```

Makefile.pas (To transfer a ATLAS*GRAPHICS data-file to FACTFILE and LONGNAME, readable by VP-Expert).

```
program makefile;
{                                                              }
{             MAKEFILE version 1.20                            }
{             (c) B.J.Kobben   1988                            }
{                                                              }
{  Converts a ATLAS*GRAPHICS data-file (.dat & .fmt) into      }
{  two VPEXPERT-readable files:                                }
{  FACTFILE - containing the regionnames and the correspon-    }
{             ding data of the chosen variable                 }
{  LONGNAME - containing the full 80-character names of the    }
{             variables                                        }
{                                                              }
{--------------------------------------------------------------}
{--------------------------------------------------------------}
{------------- DECLARATIONS -----------------------------------}
const maxnumvariables = 23;
      maxnumrecs = 250;
      datalength = 17;
      longvariablenamelength = 76;
      cnf = ' CNF 100';
      OK_char = '+';
      not_OK_char = '-';

type  action = (read,write);
      names = string[longvariablenamelength];
      data = string[datalength];
      regpack = record
                  ax,bx,cx,dx,bp,si,di,ds,es,flags : integer
                end;


var   opened, chosen : boolean;
      workstring, filename, AG_filename, feature_description,
        vpxdir, datadir : names;
      longworkstring : string[255];
      ch : char;
      commapos, quotspos : integer;
      workvariable, numvariables, numrecs, total_numrecs : integer;
      longvariablename : array[1..maxnumvariables] of names;
      variablename : array[0..maxnumvariables] of data;
      regionname : array[1..maxnumrecs] of data;
      variabledata : array[1..maxnumrecs] of data;
      count, i, j, high, norm, back : integer;
      workfile : text;

{--------------------------------------------------------------}
{---------- FUNCTIONS & PROCEDURES ----------------------------}
```

```
{-------------------------------------------------------}

procedure bliep;

begin
  write(chr(7))
end;

{-------------------------------------------------------}
procedure intro;

begin
  bliep; textcolor(high);
  write('Which ATLAS*GRAPHICS datafile do you wish to use (no exten-
sion): ');
  readln(filename);
  AG_filename := filename;
end;

{-------------------------------------------------------}
procedure wait;

begin
  writeln('Press any key to continue...');read(kbd,ch);
end;

{-------------------------------------------------------}
function no_data : boolean;

begin
  if (variabledata[numrecs] = '') or
     (variabledata[numrecs] = ' ') or
     (variabledata[numrecs] = '  ') or
     (variabledata[numrecs] = 'E+36') or
     (variabledata[numrecs] = '1E+36') or
     (variabledata[numrecs] = ' 1E+36') or
     (variabledata[numrecs] = '  1E+36') or
     (variabledata[numrecs] = '   1E+36') or
     (variabledata[numrecs] = '1E+36 ') or
     (variabledata[numrecs] = ' 1E+36 ') or
     (variabledata[numrecs] = '  1E+36 ') or
     (variabledata[numrecs] = '   1E+36 ') then
       begin
       variabledata[numrecs] := 'no_data';
       no_data := true;
       end
     else
       no_data := false;
end;

{-------------------------------------------------------}
procedure openfile(name : names; what: action);

begin
  name := name;
  assign(workfile,name);
  if what = read then
    {$I-} reset(workfile) {$I+}
  else
    {$I-} rewrite(workfile); {$I+}
  if ioresult = 0 then
    begin
    opened := true;
    end
  else
    begin
    opened := false; bliep;bliep; bliep;
    writeln(name,' can''t be opened !!');
    end;
end;

{-------------------------------------------------------}
procedure closefile(name : names);

begin
  close(workfile);
```

```
end;

{-------------------------------------------------------}
function monochrome : boolean;

var regs : regpack;

begin
  intr(17,regs);
  if (regs.ax and $0030) = $30 then monochrome := true
  else monochrome := false;
end;

{-------------------------------------------------------}
procedure initialise;

begin
  gotoxy(1,25);
  if monochrome then
    begin
    high := 15; norm := 7; back := 0;
    end
  else
    begin
    high := 15; norm := 1; back := 2;
    end;
  textbackground(back); textcolor(norm);
  chosen := false;
  opened := false;
  for i := 1 to maxnumvariables do longvariablename[i] := 'EMPTY';
  for i := 1 to maxnumrecs do variabledata[i] := '';
  filename := 'showdir.bat';
  openfile(filename,read);
  if opened then
    begin
    readln(workfile,workstring); {getting rid of "echo off"...}
    readln(workfile,vpxdir);
    delete(vpxdir,1,4);
    readln(workfile,datadir);
    delete(datadir,1,4);
    end;
  closefile(filename);
end;

{-------------------------------------------------------}
procedure ask_redefine(add_not : boolean);

begin
  bliep;gotoxy(1,24);clreol;
  write('- Variable has previously been tagged (as being ');
  if add_not then write('not ');
  write('appropiate for a choropleth)');
  gotoxy(1,25);clreol;write('Continue anyway (y/n) ? ');
  read(kbd,ch);
  if (ch = 'n') or (ch = 'N') then chosen := false;
end;

{-------------------------------------------------------}
procedure change_spaces(var changestring : data);

var temp : data;

begin
  temp := '                    ';
  for j := 1 to length(changestring) do
    begin
    ch := changestring [j];
    if ch = ' ' then
      temp[j] := '_'
    else
      temp[j] := changestring [j];
    end;
  changestring := copy(temp,1,length(changestring));
end;

{-------------------------------------------------------}
```

```
procedure read_variables;

begin
  filename := datadir + '\' + filename + '.fmt';
  openfile(filename,read);
  if opened then
    begin
    numvariables := 0;
    if not EOF(workfile) then
      begin
      readln(workfile,workstring);
      feature_description :=
        copy(workstring,2,length(workstring)-2);
      end;
    while not EOF(workfile) do
      begin
      numvariables := numvariables + 1;
      readln(workfile,workstring);
      longvariablename[numvariables] :=
        copy(workstring,2,length(workstring)-2);
      end;
    end;
  closefile(filename);
end;

{-------------------------------------------------------}
procedure read_data;

begin
  delete(filename,length(filename)-3,4);
  filename := filename + '.dat';
  openfile(filename,read);
  numrecs := 0; total_numrecs := 0;
  if opened then
    begin
    while not EOF(workfile) do
      begin
      if numrecs < 28 then
        begin
        numrecs := numrecs + 1;
        total_numrecs := total_numrecs + 1;
        readln(workfile,longworkstring);
        longworkstring := longworkstring + ',';
        delete(longworkstring,1,1);
        quotspos := pos('"',longworkstring);
        regionname[numrecs] := copy(longworkstring,1,quotspos-1);
        if (regionname[numrecs] = '') or
           (regionname[numrecs] = ' ') or
           (regionname[numrecs] = '  ') then
          regionname[numrecs] := '<Curve or Point>';
        delete(longworkstring,1,quotspos+1);
        for i := 1 to numvariables do
          begin
          commapos := pos(',',longworkstring);
          if i = workvariable then
            begin
            variabledata[numrecs] :=
              copy(longworkstring,1,commapos-1);
            if no_data then variabledata[numrecs] := 'no_data';
            end;
          delete(longworkstring,1,commapos+1);
          end;
        end
      else
        begin
        total_numrecs := total_numrecs + 1;
        readln(workfile,longworkstring);
        end;
      end;
    end;
  closefile(filename);
  writeln('number of records = ',total_numrecs);
end;

{-------------------------------------------------------}
procedure write_fact_file;
```

```
begin
  filename := vpxdir + '\factfile';
  openfile(filename,write);
  writeln(workfile,'file_read = yes',cnf);
  writeln(workfile,'datadir = ',datadir,cnf);
  writeln(workfile,'norm_color = ',norm,cnf);
  writeln(workfile,'AG_data-file = ',AG_filename,cnf);
  writeln(workfile,'workvariable = ',workvariable,cnf);
  writeln(workfile,'number_of_recs = ',total_numrecs,cnf);
  writeln(workfile,'number_of_variables =',numvariables,cnf);
  variablename[workvariable] := copy(longvariablename[workvaria-
ble],1,datalength);
  change_spaces(variablename[workvariable]);
  writeln
  (workfile,'variablename = ',variablename[workvariable],cnf);
  for i := 1 to numrecs do
    begin
    change_spaces(regionname[i]);
    writeln(workfile,'regionname[',i,'] = ',regionname[i],cnf);
    writeln(workfile,'variabledata[',i,'] = ',variabledata[i],cnf);
    end;
  closefile(filename);
end;

{-------------------------------------------------------------}
procedure write_longvariablename;

begin
  filename := vpxdir + '\longname';
  openfile(filename,write);
  writeln(workfile,longvariablename[workvariable]);
  writeln(workfile,feature_description);
  for i := 1 to numvariables do
    writeln(workfile,longvariablename[i]);
  closefile(filename);
end;

{-------------------------------------------------------------}
procedure choose_variable;

begin
  clrscr;
  writeln('Variables in ATLAS*GRAPHICS datafile ',filename);
  for i := 1 to numvariables do
    begin
    textcolor(high); if i < 10 then write(' ');
    write(i,': ');
    textcolor(norm); writeln(longvariablename[i]);
    end;
  textcolor(high); gotoxy(1,25);write('Variable to work with: ');
  bliep; read(workvariable);
  if (workvariable <= numvariables) and (workvariable > 0) then
    begin
    chosen := true;
    if (longvariablename[workvariable][1] = OK_char) then
      ask_redefine(false);
    if (longvariablename[workvariable][1] = not_OK_char) then
      ask_redefine(true);
    end
  else
    begin
    bliep; chosen := false;
    end;
  writeln;
end;

{-------------------------------------------------------------}
{---------------- MAIN -------------------------------------}
{-------------------------------------------------------------}

begin
  initialise;
  intro;
  read_variables;
  if opened then
```

```
    begin
    if opened then
      begin
      while not chosen do choose_variable;
      read_data;
      write_fact_file;
      write_longvariablename;
      end;
    end;
end.
```

Update.pas (To tag ATLAS*GRAPHICS data-variables according to the findings of CHORO-EXPERT).

```
program  update;
{                                                              }
{              UPDATE version 1.1                              }
{              (c) B.J.Kobben 1988                             }
{                                                              }
{   Updates the chosen variable of a ATLAS*GRAPHICS data-      }
{   file. If the CHORO-EXPERT system found this                }
{   variable to be appropiate for a choropleth map the         }
{   variablename in the ATLAS*GRAPHICS file .fmt will be       }
{   tagged with a + , if not with a - .                        }
{   The CHORO-EXPERT system communicates with UPDATE           }
{   through the file TEMPFACT.                                 }
{                                                              }
{--------------------------------------------------------------}
{------------- DECLARATIONS -----------------------------------}
{--------------------------------------------------------------}
const maxnumvariables = 23;
      namelength = 80;
      OK_char = '+';
      not_OK_char = '-';

type  action = (read,write);
      names = string[namelength];
      regpack = record
                  ax,bx,cx,dx,bp,si,di,ds,es,flags : integer
                end;

var   opened, chosen : boolean;
      filename, AG_filename, workstring, choropleth_OK,vpxdir,datadir
       : names;
      ch : char;
      workvariable, number_of_variables : integer;
      variablename : array[0..maxnumvariables] of names;
      count, i, j, high, norm, back : integer;
      workfile : text;

{--------------------------------------------------------------}
{----------- FUNCTIONS & PROCEDURES ---------------------------}
{--------------------------------------------------------------}
procedure bliep;

begin
  write(chr(7))
end;

{--------------------------------------------------------------}
procedure wait;

begin
  writeln('Press any key to continue...');read(kbd,ch);
end;


{--------------------------------------------------------------}
function not_done : boolean;

begin
  if (variablename[i][2] = OK_char) or
     (variablename[i][2] = not_OK_char) then
    not_done := false
  else
    not_done := true;
end;

{--------------------------------------------------------------}
function yes : boolean;

begin
  if (ch = 'y') or (ch = 'Y') then
    yes := true
  else
    yes := false;
```

```
end;

{-------------------------------------------------------------}
procedure openfile(name : names; what: action);

begin
  assign(workfile,name);
  if what = read then
    {$I-} reset(workfile) {$I+}
  else
    {$I-} rewrite(workfile); {$I+}
  if ioresult = 0 then
    begin
    opened := true;
    end
  else
    begin
    opened := false; bliep;bliep;bliep;
    writeln(name,' can''t be opened !!' );
    end;
end;

{-------------------------------------------------------------}
procedure closefile(name : names);

begin
  close(workfile);
end;

{-------------------------------------------------------------}
procedure erasefile(name : names);

begin
  assign(workfile,name);
  erase(workfile);
end;



{-------------------------------------------------------------}
function monochrome : boolean;

var regs : regpack;

begin
  intr(17,regs);
  if (regs.ax and $0030) = $30 then monochrome := true
  else monochrome := false;
end;

{-------------------------------------------------------------}
procedure initialise;

begin
  gotoxy(1,25);
  if monochrome then
    begin
    high := 15; norm := 7; back := 0;
    end
  else
    begin
    high := 15; norm := 1; back := 2;
    end;
  textbackground(green); textcolor(norm);
  chosen := false;
  opened := false;
  filename := 'showdir.bat';
  openfile(filename,read);
  if opened then
    begin
    readln(workfile,workstring); {getting rid of "echo off"...}
    readln(workfile,vpxdir);
    delete(vpxdir,1,4);
    readln(workfile,datadir);
```

```
    delete(datadir,1,4);
    end;
  closefile(filename);
end;

{-------------------------------------------------------}
procedure place_tag;

begin
  if choropleth_OK = 'yes' then insert(OK_char + ' ',variablena-
me[i],2);
  if choropleth_OK = 'no'  then insert(not_OK_char + ' ',variablena-
me[i],2);
end;

{-------------------------------------------------------}
procedure replace_tag;

begin
  if ((choropleth_OK = 'yes') and (variablename[i][2] = OK_char)) or

  ((choropleth_OK = 'no') and (variablename[i][2] = not_OK_char))
then
     begin
     {Already correct, continue without changing...}
     end
   else
     begin
     bliep;clrscr; textcolor(high);
     write(' - Variable has previously been tagged as being');
     if variablename[i][2] = not_OK_char then write(' not');
     writeln(' appropiate for a choropleth');
     write('    Do you really want to change this (y/n) ? ');re-
ad(kbd,ch);
     if yes then
       begin
       if choropleth_OK = 'yes' then variablename[i][2] := OK_char;
       if choropleth_OK = 'no'  then
                                  variablename[i][2] := not_OK_char;
       end;
     end;
end;

{-------------------------------------------------------}
procedure read_tempfacts;

begin
  filename := vpxdir + '\tempfact.';
  openfile(filename,read);
  if opened then
    begin
    readln(workfile,datadir);
    readln(workfile,AG_filename);
    readln(workfile,number_of_variables);
    readln(workfile,workvariable);
    readln(workfile,choropleth_OK);
    end;
  closefile(filename);
end;

{-------------------------------------------------------}
procedure update_file;

begin
  filename := datadir + '\' + AG_filename + '.fmt';
  openfile(filename,read);
  if opened then
    begin
    for i := 0 to number_of_variables do
      begin
      readln(workfile,variablename[i]);
      if (i = workvariable) then
        begin
        if not_done then
          place_tag
        else
```

```
          replace_tag;
        end;
      end;
    if i <= 0 then
      begin
      bliep;
      writeln(' --- ',filename,' appears to be empty !!' );
      wait;
      opened := false;
      end;
    if opened then
      begin
      closefile(filename);
      erasefile(filename);
      openfile(filename,write);
      for j := 0 to number_of_variables do
        begin
        writeln(workfile,variablename[j]);
        end;
      end;
    end;
  closefile(filename);
end;

{-------------------------------------------------------}
procedure erase_tempfacts;

begin
  filename := vpxdir + '\' + 'tempfact.';
  erasefile(filename);
end;

{-------------------------------------------------------}
{------------------ MAIN -------------------------------}
{-------------------------------------------------------}
begin
  clrscr;
  initialise;
  read_tempfacts;
  if opened then
    begin
    update_file;
    erase_tempfacts;
    end;
end.
```

Adapt.pas (To adapt an ATLAS*GRAPHICS data-file. It divides the variable chosen in VP-EXPERT by the divider chosen and if necessary multiplies it too).

```
program  adapt;
{-------------------------------------------------------------}
{              ADAPT version 1.2                              }
{            (c) B.J.Kobben 1988                              }
{                                                             }
{  Adapts the variable in an ATLAS*GRAPHICS data-file.        }
{  file. If the CHORO-EXPERT system found this variable       }
{  not to be appropiate for a choropleth map, due to the      }
{  data not being ratios, this program will change            }
{  it to ratios.                                              }
{  The CHORO-EXPERT system communicates with ADAPT            }
{  through the file TEMPFACT.                                 }
{                                                             }
{-------------------------------------------------------------}
{------------- DECLARATIONS ----------------------------------}
{-------------------------------------------------------------}
const maxnumvariables = 23;
      maxnumrecs = 100;
      datalength = 20;
      longvariablenamelength = 80;
      cnf = ' CNF 100';
      OK_char = '+';
      not_OK_char = '-';

type  regpack = record
                  ax,bx,cx,dx,bp,si,di,ds,es,flags : integer
                end;
      action = (read,write,append);
      names = string[longvariablenamelength];
      data = string[datalength];

var  opened, chosen : boolean;
     workstring, filename, AG_filename, new_filename, vpxdir, dat-
adir : names;
     registers : regpack;
     ch : char;
     longworkstring : string[255];
     dataline : array[1..maxnumrecs] of string[255];
     commapos, quotspos : integer;
     chosen_variable, number_of_variables, divide_variable, numrecs
      : integer;
     longvariablename : array[1..maxnumvariables] of names;
     variablename : array[0..maxnumvariables] of data;
     regionname : array[1..maxnumrecs] of data;
     chosen_variabledata : array[1..maxnumrecs] of data;
     divide_variabledata : array[1..maxnumrecs] of data;
     new_variabledata : array[1..maxnumrecs] of data;
     chosen_value : array[1..maxnumrecs] of real;
     divide_value : array[1..maxnumrecs] of real;
     new_value : array[1..maxnumrecs] of real;
     i,j,count, high, norm, back : integer;
     multiplier : real;
     workfile : text;


{-------------------------------------------------------------}
{---------- FUNCTIONS & PROCEDURES ---------------------------}
{-------------------------------------------------------------}
function yes : boolean;

begin
  if (ch = 'y') or (ch = 'Y') then
    yes := true
  else
    yes := false;
end;

{-------------------------------------------------------------}
function toetsstatus (var ch : char) : boolean;

begin
```

```
  registers.ax := $0600;
  registers.dx := 255;
  msdos(registers);
  ch := chr(registers.ax);
  toetsstatus := ch <> chr(0)
end;

{-------------------------------------------------------}
procedure bliep;

begin
  write(chr(7))
end;

{-------------------------------------------------------}
procedure wait;

begin
  writeln('Press any key to continue...');read(kbd,ch);
end;

{-------------------------------------------------------}
procedure cursor_on;

begin
  with registers do
    begin
      ax := $0100;
      cx := $0607
    end;
  intr(16,registers);
end;

{-------------------------------------------------------}
procedure cursor_off;

begin
  with registers do
    begin
      ax := $0100;
      cx := $2000;
    end;
  intr(16,registers);
end;

{-------------------------------------------------------}
procedure getstring  (    x,y    : integer;
                      var xstring : names;
                          maxlen  : integer;
                         capslock : boolean;
                            fill  : boolean);

var i,j          : integer;
    cursor       : char;
    space        : char;
    cleanstring  : string[80];
    workstring   : string[80];
    printable    : set of char;
    small_print  : set of char;
    cr           : boolean;


begin
  cursor_off;
  printable := [' '..'}'];
  small_print := ['a'..'z'];
  cursor := '>'; space := ' ';
  cr := false;
  fillchar(cleanstring,sizeof(cleanstring),space);
  cleanstring[0] := chr(maxlen);
  if length(xstring) > maxlen then xstring[0] := chr(maxlen);
  workstring := xstring;
  gotoxy(x+1,y); write(cursor);
  repeat
    while not toetsstatus(ch) do begin {nothing} end;
    if ch in printable then
```

```
      if length(workstring) >= maxlen then bliep else
        begin
          if ch in small print then
            if capslock then ch := chr(ord(ch)-32);
          workstring := concat(workstring,ch);
          gotoxy(x+1,y); write(workstring);
          if length(workstring) < maxlen then write(cursor);
        end
    else
      case ord(ch) of
        8,127 : if length(workstring) <= 0 then bliep else
                  begin
                    delete(workstring,length(workstring),1);
                    gotoxy(x+1,y); write(workstring,cursor);
                    if length(workstring)<maxlen-1 then
                       write(space);
                  end;
        13    : cr := true;
        24    : begin
                    gotoxy(x+1,y); write(cleanstring);
                    workstring := '';
                  end;
        else bliep
    end;
  until cr;
  gotoxy(x+1,y); write(cleanstring);
  gotoxy(x+1,y); write(workstring);
  if cr then
    begin
      xstring := workstring;
      if fill then
        for i := length(xstring) to maxlen do xstring:=xstring + '';
     end;
  cursor_on;
end;

{-------------------------------------------------------------}
procedure openfile(name : names; what: action);

begin
  assign(workfile,name);
  if what = read then
    {$I-} reset(workfile) {$I+}
  else if what = write then
    {$I-} rewrite(workfile) {$I+}
  else
    {$I-} append(workfile); {$I+}
  if ioresult = 0 then
    begin
    opened := true;
    end
  else
    begin
    opened := false; bliep;bliep;bliep;
    writeln(name,' can''t be opened !!');
    end;
end;

{-------------------------------------------------------------}
procedure closefile(name : names);

begin
  close(workfile);
end;

{-------------------------------------------------------------}
procedure erasefile(name : names);

begin
  assign(workfile,name);
  erase(workfile);
end;

{-------------------------------------------------------------}
function monochrome : boolean;
```

```
var regs : regpack;

begin
  intr(17,regs);
  if (regs.ax and $0030) = $30 then monochrome := true
  else monochrome := false;
end;

{-------------------------------------------------------------}
procedure initialise;

begin
  gotoxy(1,25);
  if monochrome then
    begin
    high := 15; norm := 7; back := 0;
    end
  else
    begin
    high := 15; norm := 1; back := 2;
    end;
  textbackground(green); textcolor(norm);
  chosen := false;
  opened := false;
  filename := 'showdir.bat';
  openfile(filename,read);
  if opened then
    begin
    readln(workfile,workstring); {getting rid of "echo off"...}
    readln(workfile,vpxdir);
    delete(vpxdir,1,4);
    readln(workfile,datadir);
    delete(datadir,1,4);
    end;
  closefile(filename);
end;

{-------------------------------------------------------------}
procedure read_tempfacts;

begin
  filename := vpxdir + '\tempfact.';
  openfile(filename,read);
  if opened then
    begin
    readln(workfile,AG_filename);
    readln(workfile,number_of_variables);
    readln(workfile,chosen_variable);
    readln(workfile,divide_variable);
    end;
  closefile(filename);
end;

{-------------------------------------------------------------}
procedure loose_spaces(var changestring : data);

var temp : data;
    t : integer;
    ch : char;

begin
  temp := '                    ';
  j := 1; t := 0;
  while j <= length(changestring) do
    begin
    ch := changestring[j];
    if ch <> ' ' then
      begin
      t := t + 1;
      temp[t] := changestring [j];
      end;
    j := j + 1;
    end;
  changestring := copy(temp,1,t);
end;
```

```
{-------------------------------------------------------}
procedure read_data;

begin
  numrecs := 0;
  while not EOF(workfile) do
    begin
    numrecs := numrecs + 1;
    readln(workfile,longworkstring);
    longworkstring := longworkstring + ',';
    delete(longworkstring,1,1);
    quotspos := pos('"',longworkstring);
    regionname[numrecs] := copy(longworkstring,1,quotspos-1);
    delete(longworkstring,1,quotspos+1);
    for i := 1 to number_of_variables do
      begin
      commapos := pos(',',longworkstring);
      if i = chosen_variable then
        begin
        chosen_variabledata[numrecs] :=
          copy(longworkstring,1,commapos-1);
        loose_spaces(chosen_variabledata[numrecs]);
        val(chosen_variabledata[numrecs],chosen_value[numrecs],
            count);
        end;
      if i = divide_variable then
        begin
        divide_variabledata[numrecs] :=
          copy(longworkstring,1,commapos-1);
        loose_spaces(divide_variabledata[numrecs]);
        val(divide_variabledata[numrecs],divide_value[numrecs],
          count);
        end;
      delete(longworkstring,1,commapos+1);
      end;
    end;
  writeln('number of recs = ',numrecs);
end;

{-------------------------------------------------------}
procedure divide;

begin
  clrscr; writeln; textcolor(high);
  writeln('YOU ARE DIVIDING'); textcolor(norm); writeln(longvaria-
blename[chosen_variable]);
  textcolor(high); writeln('BY'); textcolor(norm); writeln(longvari-
ablename[divide_variable]);
  textcolor(high); writeln; writeln('Be careful!!');
  writeln('Do you have to multiply the value of the first variable
to get the REAL value');
  write('eg. when the value is in thousands (y/n) ? ');
  read(kbd,ch);
  writeln;
  if yes then
    begin
    write('Multiply with: '); readln(multiplier);
    end
  else
    multiplier := 1;
  for i := 1 to numrecs do
    begin
    if chosen_value[i] <> 1E+36 then {if no NO DATA...}
      begin
      new_value[i] := (chosen_value[i] * multiplier) / divide_valu-
e[i];
      if new_value[i] <= 99999999.999 then          {if length <= 12
places }
        str(new_value[i]:12:3,new_variabledata[i])  {then decimal
form }
      else
        str(new_value[i],new_variabledata[i]);      {else scientific
notation}
      loose_spaces(new_variabledata[i]);
      new_variabledata[i] := ' ' + new_variabledata[i];
      end
```

```
    else
      new_variabledata[i] := '  1E+36'
    end;
end;

{-------------------------------------------------------}
procedure read_file;

begin
  filename := datadir + '\' + AG_filename + '.dat';
  openfile(filename,read);
  if opened then
    read data;
  closefile(filename);
  filename := vpxdir + '\longname.';
  openfile(filename,read);
  if opened then
    begin
    readln(workfile,workstring); readln(workfile,workstring);
        {get rid of chosen_variablename and feature_description...}

    for i := 1 to number_of_variables do readln(workfile,longvaria-
blename[i]);
    end;
  closefile(filename);
end;

{-------------------------------------------------------}
procedure get_new_variablename;

begin
  clrscr;
  bliep; gotoxy(1,2); textcolor(high); write('Please name the new
variable:');
  gotoxy(1,4); write('Variablename     :');
  gotoxy(20,4); write('Description of variable');
  gotoxy(1,5); textbackground(white); textcolor(black);
  write('                    :
                ');
  workstring := '';
  getstring(0,5,workstring,17,true,true);
  variablename[number_of_variables] := workstring; workstring := '';

  getstring(19,5,workstring,58,false,false);
  longvariablename[number_of_variables] :=
     variablename[number_of_variables] + ': ' + workstring;
  textbackground(green); textcolor(norm); writeln; writeln;
end;

{-------------------------------------------------------}
procedure rewrite_file;

begin
  number_of_variables := number_of_variables + 1;
  get_new_variablename;
  filename := datadir + '\' + AG_filename + '.fmt';
  openfile(filename,append);
  if opened then
    begin
    writeln
     (workfile,'"+ ',longvariablename[number_of_variables],'"');
    end;
  closefile(filename);
  filename := datadir + '\' + AG_filename + '.dat';
  openfile(filename,read);
  if opened then
    for i := 1 to numrecs do
      begin
      readln(workfile,dataline[i]);
      dataline[i] := dataline[i] + ',' + new_variabledata[i];
      end;
    closefile(filename);
    erasefile(filename);
    openfile(filename,write);
    if opened then
      for i := 1 to numrecs do writeln(workfile,dataline[i]);
```

```
    closefile(filename);
end;

{-------------------------------------------------------}
procedure write_new_file;

begin
  bliep; clrscr;
  writeln('Current datafile (',AG_filename,') is full !');
  write('Please give filename for new datafile (NO EXTENSION): ');
  readln(new_filename);
  number_of_variables := 1;
  filename := datadir + '\' + AG_filename + '.fmt';
  openfile(filename,read);
  if opened then
    readln(workfile,longworkstring);
  closefile(filename);
  get_new_variablename;
  filename := datadir + '\' + new_filename + '.fmt';
  openfile(filename,write);
  if opened then
    begin
    writeln(workfile,longworkstring); {region description}
    writeln(workfile,'"+',
      longvariablename[number_of_variables],'"');
    end;
  closefile(filename);
  filename := datadir + '\' + new_filename + '.dat';
  openfile(filename,write);
  if opened then
    for i := 1 to numrecs do
      writeln(workfile,'"',regionname[i],'",',new_variabledata[i]);

  closefile(filename);
end;

{-------------------------------------------------------}
procedure update_file;

begin
  if number_of_variables < 22 then
    begin
    rewrite_file;
    end
  else
    begin
    write_new_file;
    end;
end;

{-------------------------------------------------------}
{---------------- MAIN ---------------------------------}
{-------------------------------------------------------}
 begin
  initialise;
  read_tempfacts;
  if opened then
    begin
    read_file;
    divide;
    update_file;
    end;
end.
```

Install.pas (to install the system. The proper directories for the CHORO-EXPERT system and the ATLAS*GRAPHICS data-files are stored in two REM(ark) lines in showdir.bat for future reference by the pascal programs).

```
program install;
{-------------------------------------------------------------}
{------------- DECLARATIONS ----------------------------------}
{-------------------------------------------------------------}
const namelength = 80;
      blank = '                                ';

type  action = (read,write);
      names = string[namelength];

var   opened, stop : boolean;
      workstring, filename, vpxdir, datadir, vpxdrive, datadrive :
names;
      ch : char;
      no : integer;
      workfile : text;

{-------------------------------------------------------------}
{----------- FUNCTIONS & PROCEDURES --------------------------}
{-------------------------------------------------------------}
procedure bliep;

begin
  write(chr(7))
end;


{-------------------------------------------------------------}
procedure openfile(name : names; what: action);

begin
  assign(workfile,name);
  if what = read then
    {$I-} reset(workfile) {$I+}
  else
    {$I-} rewrite(workfile); {$I+}
  if ioresult = 0 then
    begin
    opened := true;
    writeln(name,' opened...');
    end
  else
    begin
    opened := false;
    writeln(name,' can''t be opened !!');
    writeln(name,' must be in current directory !!');
    end;
end;

{-------------------------------------------------------------}
procedure closefile(name : names);

begin
  close(workfile);
  writeln(filename,' closed...');
end;

{-------------------------------------------------------------}
procedure initialise;

begin
  stop := false;
  opened := false;
  filename := 'showdir.bat';
  openfile(filename,read);
  if opened then
    begin
    readln(workfile,workstring); {getting rid of "echo off"...}
    readln(workfile,vpxdir);
    delete(vpxdir,1,4);
```

```
    readln(workfile,datadir);
    delete(datadir,1,4);
    end;
  closefile(filename);
end;

{-------------------------------------------------------}
procedure kader;

const lig_streep = '═'; sta_streep = '║';
      lb_hoek = '╔'; rb_hoek = '╗';
      lo_hoek = '╚'; ro_hoek = '╝';

var y,x : integer;

begin
  normvideo;
  y := 2; x := 1;
  gotoxy(x,y);write(lb_hoek);
  for x := 2 to 79 do
    begin
    gotoxy(x,y);write(lig_streep);
    end;
  x := 80; gotoxy(x,y);write(rb_hoek);
  for y := 3 to 23 do
    begin
    x := 1; gotoxy(x,y);write(sta_streep);
    x := 80; gotoxy(x,y);write(sta_streep);
    end;
  y := 24; x := 1;
  gotoxy(x,y);write(lo_hoek);
  for x := 2 to 79 do
    begin
    gotoxy(x,y);write(lig_streep);
    end;
  x := 80; gotoxy(x,y);write(ro_hoek);
  lowvideo;
end;


{-------------------------------------------------------}
procedure write_batch;

begin
  filename := vpxdir + '\showdir.bat';
  openfile(filename,write);
  if opened then
    begin
    writeln(workfile,'echo off');
    writeln(workfile,'rem ',vpxdir);
    writeln(workfile,'rem ',datadir);
    writeln(workfile,'copy nofacts.vpx factfile.');
    writeln(workfile,'if exist longname. del longname.');
    writeln(workfile,'if exist tempfact. del tempfact.');
    writeln(workfile,'cls');
    writeln(workfile,'dir/w ',datadir,'\*.dat');
    writeln(workfile,vpxdir,'\makefile');
    writeln(workfile,'echo on');
    end;
  closefile(filename);
end;

{-------------------------------------------------------}
procedure choro_path;

begin
  gotoxy(50,7); write(blank);
  gotoxy(50,7); read(vpxdir);
end;

{-------------------------------------------------------}
procedure data_path;

begin
  gotoxy(50,8); write(blank);
  gotoxy(50,8); read(datadir);
```

```
end;
{-------------------------------------------------------------}
procedure save_and_quit;

begin
  clrscr;
  write_batch;
  stop := true;
end;
{-------------------------------------------------------------}
procedure menu;

begin
  clrscr;
  kader;
  normvideo;
  textbackground(15); textcolor(0);
  gotoxy(26,3);writeln('   CARTO-EXPERT INSTALL MENU  ');
  textbackground(0); textcolor(15);
  lowvideo; gotoxy(10,7); write('Current path for CARTO-EXPERT
system: ');
  normvideo; gotoxy(50,7); write(vpxdir);
  lowvideo; gotoxy(4,8); write('Current path for ATLAS*GRAPHICS
data-files: ');
  normvideo; gotoxy(50,8); write(datadir);
  lowvideo;
  gotoxy(13,15); writeln('1 - Change path for CHORO-EXPERT system');
  gotoxy(13,16); writeln('2 - Change path for ATLAS*GRAPHIC
    datafiles');
  gotoxy(13,17); writeln('3 - Save changes and quit');
  gotoxy(13,18); writeln('4 - Quit without saving changes');
  gotoxy(13,13); write('Choose: '); read(kbd,ch);
  normvideo;
  case ch of
    '1' :  choro_path;
    '2' :  data_path;
    '3' :  save_and_quit;
    '4' :  stop := true
  else
    begin
    bliep; bliep; bliep;
    stop := false;
    end;
  end;
end;

{-------------------------------------------------------------}
{------------------ MAIN -------------------------------------}
{-------------------------------------------------------------}

begin
  clrscr;
  initialise;
  while not stop do menu;
  clrscr;
end.
```