$1^{-}3^{?}$ ...

THE DEVELOPMENT OF A NATURAL LANGUAGE INTERFACE TO A GEOGRAPHICAL INFORMATION SYSTEM

S4-82
150 . 98
N93-22453

Sue Walker Toledo, Ph.D.
Senior Scientist, Netrologic
5080 Shoreham Place, Suite 201
San Diego, Ca. 92122

Bruce Davis, Ph.D.
Project Manager
NASA at the John C. Stennis Space Center
Stennis Space Center, Ms. 39529-6000

November 27, 1992

Introduction

This paper will discuss a two and a half year long project undertaken to develop an English-language interface for the geographical information system GRASS. The work was carried out for NASA by a small business, Netrologic, based in San Diego, California, under Phase I and II Small Business Innovative Research contracts. We consider here the potential value of this system whose current functionality addresses numerical, categorical and boolean raster layers and includes the display of points sets defined by constraints on one or more layers, answers yes/no and numerical questions, and creates statistical reports. It also handles complex queries and lexical ambiguities, and allows temporarily switching to UNIX or GRASS.

The Need for More Natural Computer Interfaces

Let us first review some of the more obvious reasons for why one might want to undertake developing a natural language interface to a geographical information system (GIS). More subtle, but very important, reasons for developing simpler interfaces than we have today will come up later.

If we could find a way to enable all computer interfaces to handle our language as we do, there would be a greatly reduced time spent in learning the computer systems we employ, there would be easier communication between separate groups using the same or similar data, and we would be likely to experience less frustration in our work, and generally be more efficient in our use of our computers. The computer would suddenly become accessible to many more people, especially if good speech recognition and machine translation were standard parts of such systems. All we have to do to realize these things, if we are using computers very intensely in our work, is to think of the tomes of manuals we have had to read, and count the number of different programming languages, word processors, network protocols, etc. we have had to learn in our lifetime. It helps to imagine how long it would take to teach a high school student all we know about the computer to do our jobs ...say a sub-Saharan African high school student. Or we can think of the frustration an ARC/INFO user finds when he comes to visit a colleague who uses INTERGRAPH or GRASS or ERDAS. If we use a GIS under UNIX, the manuals for those two pieces of software alone can sometimes consume two long book shelves. In addition, we probably also have hardware, and editor or word processor (and probably C) documentation as well in the same bookcase. It seems no one any more even aspires to knowing every capability of the software systems they use, as many did twenty years ago when systems were less complex. Undoubtedly many errors are made because of the difficulty busy people have in finding the time to learn the complicated systems they must use.

Current Solutions Often Proposed to Solve the Problem: GUIs and Natural Language Interfaces

Today many say that the solution to the problem of the complexity of command line interfaces to the computer (such as DOS, UNIX, C, and the basic programming language of all GISs) are the GUIs - graphical user interfaces. It is important to make a distinction here, however. What a GUI is actually required to be is a multitasking windows environment. It is clear that such GUIs, whose windows are generally now opened and controlled through an interface that is standard across all systems, are very valuable additions to the interface scene. But what goes on in the individual windows are separate processes, each of which may have quite different interfaces.

When people say that GUIs are a solution to the interface problem, they usually have in mind not only that there is a standardized multitasking windows interface being used, but also that there is a menu interface (also following industry-wide standards in its construction) inside the control windows, involving icons and/or simple, easily understood English phrases that the user can choose between by pointing and clicking with a mouse. An excellent menu-driven GUI is also unquestionably a great improvement over a command line interface to computer software of any complexity.

Another solution to this problem that is sometimes proposed is a natural language interface. Of course one would usually want it embedded in a GUI, as ours in fact is: it includes several windows, for instance a GIS display window, and a system operation log window in which the user can watch what is going on behind the scenes if he wishes, and the basic control window, which allows the user to give English language commands or queries, or GRASS or UNIX commands, or to make other choices from a GUI-type menu. The question we want to address in this paper is how valuable the addition of the natural language interface can be.

Thus we will discuss here the relationship between three specific kinds of interfaces that can occur in any environment, 1) menu-driven interfaces, involving choices through icons, slide bars, numbers or natural language phrases, 2) natural language interfaces, receiving questions or commands in full natural language sentences, and 3) interfaces focussing on graphical interactions with the user, e.g. allowing interactions such as the user outlining a region for the system to analyze, or for a display to zoom to, or the user dragging parts of a drawing into other positions within the same drawing in a CAD system. (Note that in our terminology we are not calling any GUI a graphical interface, in spite of every GUI having simple graphical capabilities of dragging things from window to window, allowing the user to point to choices, etc. We will be using the term "graphical" only for interactions that go beyond the ones embedded in the generic GUI.) The three kinds of interfaces have overlapping, but sometimes different, advantages and disadvantages. Since a graphical interface is indispensable whenever the task is primarily graphical, and since graphical interfaces can be integrated with both menu and natural language interfaces, we will focus our attention on comparing menu-driven interfaces and natural language interfaces.

Menu-driven interfaces have the following advantages over natural language interfaces:

a) Their very structures teach the capabilities and the limits of the software to which they interface (very important). Good menu interfaces for systems of limited functionality are easy to learn, and give users a sense of confidence that they understand the system they are employing.

b) The "state-of-the-art" is more advanced, in that standards for menu construction have been established and helpful tools have been developed to make menu interface construction and maintenance easier and less time-consuming than in the past.

c) People have worked out well how to integrate menu interfaces with graphical interactions critical in applications.

d) Sometimes a menu can be structured in such a way as to force users to use the functionality efficiently.

e) Today's menu interfaces generally take much less space and work much faster than natural language interfaces.

f) Menu-driven interfaces embedded in GUIs are sometimes preferred by people who cannot type well, but can nevertheless manipulate a mouse.

However, a good natural language interface would have certain advantages over a good menu-driven interface for the same system. Here are some advantages of natural language interfaces, including some that arise in situations in which a menu-driven interface is not possible.

1) Systems of the future will be larger, and integrate many functionalities, which will mean wide and deep menu structures (thousands of icons are not an easy thing to remember, for instance; and a menu interface built up with hundreds of thousands of words or phrases is bewildering to try to grasp as a whole). In a situation of this kind of complexity, natural language interfaces will also save the user from the tedium of being forced to go down through many levels of menus, or to choose from long lists.

2) If the menu choices involve many submenus and a long succession of different requests, the user can easily loose track of what he is doing, as menus disappear from his screen and his mind begins to forget the sequence of choices he has made. Compare the situation to having a natural language query, or even a number of them, before one's eyes. The idea is that our minds seem to grasp an integrated, "streamed" task request better than one broken into many parts that have to be carried out in sequence.

3) Natural language interfaces allow single-sentence requests which cannot be directly specified through the menu-structure. To accomplish them in a menu-driven system would involve manually working out the equivalents of conditionals and loops, for example.

4) The natural language user doesn't have to think in terms of any specific structure on the overall functionality of the system. He just asks for what he wants in one of the natural ways to request some functionality. (The particular structure the menu builder has put on the functionality may be only one of several choices, and might be in conflict with the user's natural way of organizing the same material in his head.)

5) The natural language user does not suffer the same kind of disturbance when a great deal of functionality is added to the system, while the menu user can see his whole set of menus reorganized (ways of requesting things that have become automatic are now changed).

6) Ways of responding to people with different levels of education (the beat patrolman, the police detective, the chief of police) would not be evident, as in menu interfaces, where the standard procedure is to use completely different interfaces for different groups of people (so that each user can completely understand his interface).

7) Machine translation is one kind of natural language interface to a free-text (or structured) database. The need for this kind of natural language interface becomes increasingly critical as the world comes rapidly together economically and politically.

8) Only a highly developed natural language capacity in the computer will permit realistic virtual reality scenarios involving simulated human speech and the interpretation of real human speech.

9) Without a robust natural language capacity it will be hard for us to easily extract all the information we will need from the large free text databases that will become commonplace soon.

10) Natural language interfaces would very likely allow people to think more freely in their work, with the result that they might be significantly more creative and more efficient.

Strange as it may seem, more people initially resist the idea of natural language interfaces because they say they can't type, or because they can't spell, than for any other reason. (Menu enthusiasts should remember that other users claim more difficulty with mousing than with typing.) Spelling- (and grammar-) correctors and speech recognition can now eliminate such issues. The speech-recognition technology is just coming on the scene, of course, and has quite a few problems of its own, many of which are tied in with the linguistic problems that have a lot to do with what make the state of the art of constructing natural language computer interfaces less "advanced" than that of building menu interfaces.

The great advantage of the menu-driven interface is that it lays out the capabilities of the system for the user (assuming there are clear explanations behind every menu choice, accessible through the help facility of the system, which is not always the case, of course). The best that the most developed natural language systems generally do now is give the user access to a short discussion of system functionality if the user asks a question like "what can you do?" As a result the user of a natural language interface will too often ask the system to do things it has not been designed to handle. Which points out a second important advantage of a menu-driven interfaces - that presumably the system will always be able to do what you ask of it (although if you do not quite understand what you are doing nothing can prevent you from asking for the wrong thing). It is also the case that new users to natural language systems often employ natural language constructions that the linguistic side of the system cannot yet handle, i.e. the limitations on what the system can accomplish through a natural language interface go beyond the limitations of the software behind the interface.

Regarding the teaching capabilities of the two kinds of system, it is obvious that the natural language system builder can put answers to a large variety of questions about the system into his interface, but the teaching side of natural language systems is likely to be very limited for some time, while developers are spending their time working out some of the more critical linguistic problems. This true disadvantage of the natural language interface is closely connected with one of its greatest advantages, however, as I will try to explain in a moment. It would seem (for many reasons) that the ideal interface would be one that integrated all the things we do "naturally,"so that all three kinds of interfaces we are discussing would be included, and generally within the GUI multitasking windows environment. In such a system the user would have the benefit of both kinds of teaching styles just mentioned, the more discursive one that would be natural in response to natural language questions, and the structured one that the menu interface laced with good "help" automatically provides (of course there is discursive help behind each menu item as well). He could also have visual instruction, which would be necessary to teach him about many graphical interactions.

I would first like to make somewhat graphic the first two advantages claimed for a natural language interface over a menu interface. The computer systems of the future are likely to be very large - e.g. the database containing all national hospital information (one for all military personnel now exists). Consider the national spatial data infrastructure database about to come on line. The whole GIS world is attending to the data standards that will be used to make all this data simultaneously available to all the federal agencies that can make use of it. Now consider all the sciences behind the queries that will be asked of this database. The menu structure that would allow all the scientists to *easily* query such a database would be broad and deep indeed, and take a very long time to construct, moreover.

For instance the soil scientist that wishes to get information out of one of the soil layers in the database will not want to first consult a book or computer file giving the names of the layers for the different counties in the United States, then look at the layer containing his data to see the names of the soils in his region (generally on the order of fifty to a hundred different soils), then locate the soil manual for the county to find out which soils have the characteristics that he is concerned with, and finally request a map of the soils in question. He will want a menu structure that will help him do this.

For instance, if he wants to see the wetlands soils in Hancock County, Mississippi, and there is no natural language interface that permits him to just type in that request, then he will want a menu system that will allow him to specify the county he is concerned with (presumably he would first choose the state out of a list of fifty, then choose the county in that state out of a list of similar length), and then that he wants soil information, and then that he wants to see the wetlands areas. Note that if the menu structure is to allow him the choice of requesting the wetlands soils, it will also probably allow him to specify other kinds of soils of interest to soil scientists, e.g. soils with different drainage, erosion, runoff, acidity etc. properties. So this choice part of a menu can get quite complicated. Also the menu must somehow allow him to construct at least boolean constraints involving a number of different layers simultaneously, which adds more complexity.

Thus what points 1 and 2 above are getting at is that to build a menu structure that allows a user to easily query the basic data of a science means to build most of the concepts of the science into the menu structure. And since our personal screen of vision can only scan a limited number of items at once without getting lost, we must have many submenus (or allow typed input in our menu system).

The natural language query "what county in the state has the smallest number of property owners?" would presumably illustrate point 3.

I would like to add a little to the content of points 4 and 10, for they may concern factors of much greater importance than anyone now realizes. Recall that I claimed that one of the main disadvantages of a natural language interface, that a user will often ask for something the system cannot do, is connected with one of its most interesting advantages. The natural language interface user will be likely to spend more time thinking about what he wants the system to give him, about setting his goal, (for that is what one wants to specify in a natural language command or query), while the menu interface user will be more likely to think more about what he can do with the system before him, or about how the system can be used to do the particular thing he wants today. He can actually often leave his goal only partially formed and depend on the path he will be forced to take down through the menu structure to make it more concrete (since he knows he can only do what the menu allows anyway). The menu interface user generally lives intellectually within the confines of his system, and doesn't tend to waste much time questioning its limitations.

The natural language interface user is not so obviously confined, and when he asks for something the system can't yet do, he will tend to immediately think of asking developers to add new functionality. In this sense the natural language interface is an "open" interface in terms of the users' mindset, while the menu interface is a "closed" interface. Except in situations where what is expressed is most naturally expressed in images, drawings, non-language sound, or gestures, it would seem that natural language could express anything expressible in a menu interface. Certainly the expressive powers of natural language are limitless, which the expressiveness of even a command language is limited by the capacities of the hardware it runs on. In any case, it would seem that being accustomed to focusing on one's goals, the object one requires, rather than on "how to" issues, would tend to lead to more creative thought in one's work.

Related to this is that in a situation of a very complex menu structure, where doing anything of much content requires many choices being made, a robust natural language interface that would often allow the user to just simply specify what he wants done in the language that comes first to mind would be greatly appreciated by the scientists using the interface, due to the savings in time involved. Thus a high quality natural language interface might be employed a very great part of the time by highly educated users. (This requires, however, that natural language interfaces be developed in such a way that the scientist can ask for all the information he needs to know to make sure the work he is doing meets the highest standards of his science. For instance, he must be able to ask about the accuracy and precision of the data he is accessing, the dates it was gathered and by whom, precisely how it is being manipulated by the programs responding to his request.)

I would like to give a little more evidence for believing that the freedom of thought that might be more encouraged by an excellent natural language interface could significantly increase creativity in one's work. But first I want to distinguish two different kinds of GIS menu interfaces for you. One is a generic interface, designed to just let you access the basic underlying functionality of the GIS in a clearer way than through short command line function calls (for instance, for GRASS, the excellent interface put out by Osiris). The second is one that has been designed from the point of view of the tasks that the user wishes to carry out. It uses icons or the ordinary vocabulary of the user, and allows the user to specify that tasks be carried out by choosing icons or phrases that are natural for him to use. This would usually be called an application interface. The kind of natural language interface that we developed, and which I am concerned with discussing here, is also an application natural language interface.

(Note that being an "application interface" does not mean that the interface is useful in only one application, but that it is designed to be used by people in applied work. For instance, an interface developed to do environmental monitoring will have to have the vocabulary of all the many sciences that are brought in during the analysis that must be carried out. As we try to use computers to integrate more and more of the data of our society, we will want systems that have built into them more and more of the vocabulary we think in.)

Consider doing something fairly simple through one of the application menu interfaces. The soil science example a few paragraphs back already shows how the menu interface is likely to become tedious when one wants to make a few constraints involving a number of layers (and we started the example several choices down). Having to go to so much trouble to ask for something relatively simple slows the user down, interrupts his normal thought processes (introduces frustration into the situation, which tends to focus attention on the cause of the frustration and take it off the goal of the endeavor).

The situation is worse for the generic menu interface. Say you want to do something that in your GIS takes a hundred lines of commands. The situation is far better than when you didn't have the menu interface, since you now don't have to remember the precise syntax of the many commands in the underlying GIS language. But you are still going to have to execute those same lines of GIS program, just now by choosing more intuitive things from the menu rather than writing cryptic command line statements.

The same problem will hold for the application natural language interface, of course, IF the complex program in question is not using any of the application concepts. Command line programmers, menu users, and natural language users will always try to decompose any long and complex program they have to write to see if there are pieces that will be employed over and over again. These they build into macros in the first two cases (this is often not possible in menu-driven systems, however), while the natural language user will define a new concept (the equivalent of a function or macro name). And as we have more and more macros and functions in the system that a person will want to use, the choice lists for them in a menu will keep getting longer, i.e. the choice is harder to get at. As the only true language manipulating animal, however, we seem to have the capacity to be able to simultaneously remember the meanings of millions of different words/phrases, so that we shouldn't really need to be prompted. We can just say/write what we want. Menus can be built in such a way to help us get to our choice more easily, of course, even prompt us when we can only think of something related to our choice, or let us type in individual phrases. Once this is permitted, then the question again becomes which is faster and more accurate for the user, the menu, or natural language sentences.

An example I like to think about in this command level and generic GIS interface context is the following. Writing a mathematical proof in a formal logical system is very similar to writing a command line GIS program, and proofs so written out are checkable by computers for correctness. But no mathematician (not even mathematical logicians) would consider doing such a thing, because they believe that not only would it increase the time required to write a paper by a very great amount, but would also reduce their creativity to zero. They want to write their papers at the higher level they think in. (Most published papers contain correct proofs, incidentally.)

Most of what I have said so far seems to be weighted towards natural language interfaces. Why are there so few people developing or using them as database interfaces? Well, the problem is that even after thirty years of work, the current parsers, the programs that take an English language query or command and attempt to interpret it for the computer, still do not do very well much of the time. The last thirty years have involved a lot of attention to what most people call syntax, determining the part of speech of the different words in a sentence, finding the subject, the predicate, and the prepositional phrases, and then what those phrases modify. This is not an easy task, incidentally, because many words in English may be of different types of speech in different contexts.

While solving this problem (at least to a great extent; most systems will stumble over some constructions, however), linguists also learned a lot about the varied efficiencies of different methods to do this. But there is not yet any consensus about how to approach the deeper problem of resolving all the subtleties of meaning that different words and phrases can have. Here are some examples of the kinds of ambiguous words of phrases that occur constantly in natural language, making difficult the task of grasping the meaning of a sentence for a computer:

Show the soils *there.*

How many soils are *there*?

Calculate the *area* covered by ocilla soils.

Show me the *area* covered by ocilla soils.

What are the poorly drained soils?

what: list the names or display a map?

poorly drained: all of "excessively poorly drained", "moderately poorly drained" and "poorly drained", or just "poorly drained"?

if a display wanted: what colors (default colors of the SCS soil layer, white for the points in question and black elsewhere, different colors for the three kinds of poorly drained soils and black elsewhere, or some other color scheme)?

Show me the area *near* the test stand where acid deposition was less than twice the average acid deposition at the test stand today.

Show me the area *near* Chicago where acid deposition is less than twice the average acid deposition in Chicago today.

The better quality natural language interpreters today do have means built in to engage the user in dialogue about ambiguities, which is invaluable to the natural language interface user (and which is a technique we humans use to resolve ambiguities in conversation far more than we are aware of), but they do not have enough broadly applicable techniques for resolving the masses of ambiguity we resolve (or realize it is not necessary to resolve) without dialogue.

In addition to getting in trouble with the many ambiguities involved in natural language, there is another major stumbling block in the way of natural language interface developers. A natural language interpreter requires constant maintenance of its own vocabulary/database-connections-to-the-vocabulary database, unless it is dealing with a completely static database. It is this, especially when added to the fact that the systems do not always perform linguistically the way one would like them to, which has probably prevented a wider use of natural language interfaces.

The Place of Natural Language Interfaces Today

The question then is, in these early stages of the development of natural language interfaces, should someone with limited funds attempt to use this technology. Also, is the technology sufficiently important that the government and industry should finance any significant amount of research and development in this area? I tend to think the answer is yes, to both questions (in the first case, though, only for the relatively affluent user who can afford a talented person to maintain his natural language interpreter's database), and for the following reasons.

A good natural language interface would be a very valuable thing, for the many reasons discussed above, and the only way we can obtain such systems is by serious research, for which realistic natural language commands/queries is essential. It is very difficult to gather a realistic corpus of such linguistic input data without putting a system out in the field with a reasonable amount of functionality (which is possible now) and then obtaining feedback from users about what is lacking.

Far better, though, would be to get users, before directly interacting with the computer, to give to a tape recorder the commands/queries they would really like to ask the system, even though they may know already that with their current system that would not work; for in this way linguists would not only get more detailed linguistic input, but also have both natural language and speech recognition input data for the database interface research. I believe that it is only in the context of such a functional natural language interface that the kind of detailed feedback can be obtained that is needed to set priorities on the linguistic research that must be done.

Moreover, in many situations a well-developed system based on the current technology will be easier for the user to use than a menu interface devised to address the same tasks and the same data using the same set of concepts. An attempt should be made to develop fully integrated interfaces from the beginning (menu, natural language, and graphical combined), so that we can get a better feeling for the strengths and weakness of the different aspects of the integrated interface. Moreover, if natural language is to be used, it is essential to provide such interfaces now, while the state of the art in the natural language area is in the process of active development, so that users can have something easier to use than command line programming when the natural language interface is incapable of doing what they would like to ask of it. I would predict, moreover, that we would learn to construct even more flexible menu interfaces if we created some within integrated systems where the goal was to always come as close as possible to the ease of the natural language interface it is paired with. An integrated interface such as this will be essential whenever truly graphical functions are required by the system. Moreover there will always be subsets of functionality that will be much easier to do through menu interfaces than by natural language, no matter how robust natural language systems become (for instance, situations in which a small number of finite choices are involved; examples are the ATM machine, or choosing formatting fonts and margins, or specifying map colors, the latter of which can be done very elegantly with sliding bars that change the color until it is the one you want ... but don't you really wish those phone answering machines that take you down through many submenus could be replaced by a machine that could just understand when you say what you want?!).

The Netrologic Natural Language Interface to GRASS

We hope that the system we are in the process of developing, an English-language interface to the GIS GRASS, will have sufficient functionality so that it can soon be a working prototype out in the field gathering this kind of input data for further developments. Let me now tell you a little more about it. The current and near-term functionality of the interface system is the following:

Displays points sets defined by a set of constraints

Answers yes/no and numerical questions, and prints reports

Handles categorical, numerical, and boolean data

Handles ambiguity in words, e.g. "area"

Changes map windows to fixed regions, e.g. "quadrants"

Handles complex queries, including commonly used functions

Permits UNIX and GRASS commands instead of an English query

Integration of Graphical Features of GRASS, e.g. finding of coordinates and specific data values

The developers have thought about many long-term extensions of the system that would be valuable to users, for instance the inclusion of a library of functions commonly used within the GIS community, and building various expert-system features into the system. The first extension of this kind should of course be to extend the system to cover all of GRASS's basic functionality. The basic system design also endeavored to make the system amenable to modification to enable it to be serve as an interface to GISs other than GRASS.

The Basic System Design

The system was constructed by modifying PARLANCE, a commercial natural language interface to relational databases developed by BBN, so that it no longer generates SQL (the standard command language for relational databases), and then building a bridge between the meaning representation language (MRL) output of its linguistic interpreter to programs in the GRASS command language.

The basic system design can be described by the following components, through which the data successively flows:

- The PARLANCE user interface (modified)
- The PARLANCE English query interpreter (truncated)
- A transformation module, producing "SpatialFlattenedMRL"
- The SFMRL interpreter, which translates SFMRL into calls on the GRASS drivers
- GRASS drivers
- GRASS (calculations, displays, etc.)
- BBN's output handling (in the case of a textual response)

In conclusion, let me mention that the following goals for current and future developments of a natural language capacity in the computer appear to have been taken by workers in the NL/AI community. A large number of papers and even systems exist illustrating directions proposed for addressing one or another of these points. It is possible that more attention to the first one as the appropriate starting point might be helpful.

- o Gather a large representative sample set of the queries and commands people would really like to be able to make, for each discipline and task, to enable linguists to be able to do the appropriate linguistic research. Spend a large amount of time on a "preanalysis" of this data, deriving from it the most critical problems to be addressed, and a prioritization of these problems.

- o Develop an appropriate meaning representation language (MRL) into which queries and commands can be translated; if possible this should be a universal "interlingual" that would work for all the computer natural language applications, and be independent of the natural language in question.

- o Expand the linguistic community's understanding of the use of language so that the queries and commands a user would like to employ can be "parsed" into the meaning representation language in a way adequate to the user's needs (possibly only after some dialogue with the user).

- o Work out how to use common and scientific knowledge, as well as computer-generated models of individual users, for reasoning and other "expert system" purposes, as well as to aid in the parsing of queries and commands, and determine where to embed this knowledge in the NL systems that are used.

- o Develop systems that are extremely flexible, can easily change as the language changes (a very large amount of new scientific and technical vocabulary enters each natural language every year). Develop systems where the different natural language components - speech recognition, machine translation, free text database retrieval, GIS, relational, and object-oriented database interfaces, etc. - can be changed simultaneously as the state of the art in natural language interpretation develops.

- o Find a way to develop systems whose underlying structure is sufficiently easily understandable that they can be maintained without great difficulty as the language of users and the underlaying functionality of the systems change.

- o Natural language capabilities should be integrated into easily usable systems along with many other access techniques that are being developed.

- o Learn how to develop systems that can teach users their capabilities and how to use them.

You Can Say the Same Thing Many Ways in Natural Language

(And spell-correcters and speech-recognition help with the typing problem.)

display the soils
show the soil types, please
will you please show the soil layer
give me the SCS layer
display the soil data
display the data on soils
show the soils in the region
paint the soils here
display the scs layer
show the soils data
give me a soil map
map the soils
paint the soils for me
give me a map of the soils
what the hell do the soils look like
show the soils in the area
show the soils of the area

ETC.!

show the terrain between 5 feet and 10 feet in altitude
show me elevations ranging between 5 and 10 feet
show me where the elevation ranges between 5 feet and 10 feet
display the land which is between 5 feet and 10 feet in altitude
display places of elevation between 5 feet and 10 feet
display points where the elevation is larger than 5 feet and smaller than 10 feet

show any soil that is atmore or ocilla
display all soils that are atmore or ocilla
show the atmore soils and the ocilla soils

vc6

Parsing: **SHOW POORLY DRAINED SOILS WHOSE ELEVATION IS NOT GREATER THAN THE AVERAGE ELEVATION**

MRL:

```
(FOR SOME
   X.125
   /
   PNT
   I:I
   (AND
    (FOR SOME
       X.157
       /
       SOIL-CLASS
       I:I
       T
       I;I
       (AND ( I POORLY DRAINED I X.157)
          (EQ (PNT-SOIL-OF X.125) X.157)))
   (NOT
    (FOR THE
       X.153
       /
       LENGTH
       I:I
       (EQ (PNT-ELEVATION-OF X.125)
         X.153)
       I;I
       (FOR THE
          X.149
          /
          (GENERATE AVG
                X.150
                /
                PNT
                I:I
                NIL

                 I;I
                (PNT-ELEVATION-OF X.150))
          I:I
          NIL
          I;I
          (GT X.153 X.149)))))
I;I
(DISPLAY (PNT-SOIL-OF X.125)))
```

Parsing: **SHOW POORLY DRAINED SOILS WHOSE ELEVATION IS NOT GREATER THAN THE AVERAGE ELEVATION**

<u>GRASS COMMANDS and RESPONSES</u>

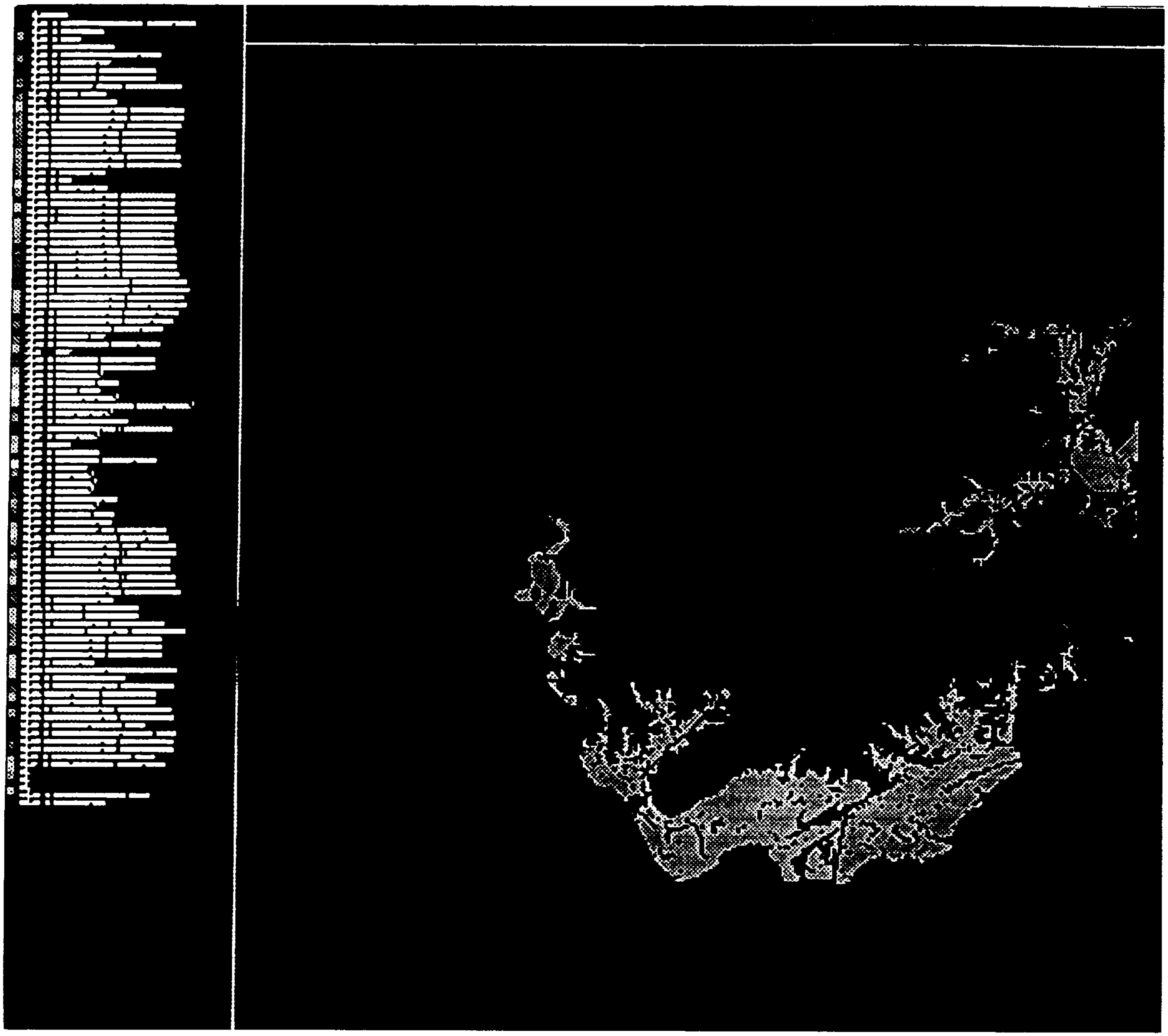
```
g.region       -d
g.remove       MASK > /dev/null
r.stats        -cm input=elev15q.20m, output=/netro/n.sum
r.stats:  complete ...  100%

Waiting for completion...


g.region       -d
g.remove       MASK > /dev/null
r.mapcalc      < /netro/n.tbl
echo           1=1  | r.reclass in=n_result out=MASK title=MASK
d.mon          select=x0
d.erase
d.colormode    fixed
d.frame        -s frame=image
d.rast         -o map=han.sol
d.frame        -s frame=legend
d.erase
d.legend       map=han.sol
d.frame        -s frame=image
d.mon          unlock=x0

PARSING EXPRESSION ...
EXECUTING n_result = ...  100%
CREATING SUPPORT FILES FOR n_result
minimum value 0, maximum value 1
expression stack size 13, execute stack size 3
```

"Show Poorly Drained Soils whose Elevation is Not Greater than the Average Elevation"



J.L. Star Aug 92