Center for Advanced Computation

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

CAC Document No. 188

IRIS/NARIS

A Geographic Information System for Planners

by

Peter A. Alsberg et al

January, 1975

CAC Document Number 188

IRIS/NARIS

A Geographic Information System for Planners

by

Peter A. Alsberg
William D. McTeer
Stewart A. Schuster

Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

January, 1975

This work was supported by
The Ford Foundation,
The Northeastern Illinois Planning Commission, and
The Illinois Institute for Environmental Planning

TABLE OF CONTENTS

Page

Page

IRIS/NARIS

A Geographic Information System for Planners

1. Introduction and Summary

The modern land use planner has to correlate and analyze large quantities of diverse data. For example, a planner will normally examine data about income, land use, soils, industry, population, housing, geology, forestry, transportation, etc. for a single site location problem. The only attribute that all this data shares is location. It all applies to a specific place on the face of the earth. Systems which use geographic locators to structure these diverse kinds of data are called geographic information systems.

Geographic information systems can answer two basic questions. First, they can tell you the attributes of a given location. Second, they can tell you in what locations a given set of attributes occur. The first question is asked by planners when checking that proposed land use modifications for a given site are compatible with that site and in conformity with the regional plan. The second question is asked when trying to choose optimal sites for locating new or relocating existing activities.

The Center for Advanced Computation of the University of Illinois at Urbana-Champaign began building the Natural Resource Information System (NARIS) in early 1970. NARIS was originally designed for the storage and analysis of natural resource data in an eight county region surrounding Chicago. NARIS was built in cooperation with the Northeast Illinois Natural Resource Service Center (a local agency), the Northeastern Illinois Planning Commission (a state agency), and was supported by funds from the Ford Foundation and several state agencies.

One year later, the Illinois Resources Information System (IRIS) Feasibility Study was begun [1]. The study was supported by three State of Illinois agencies: the Institute for Environmental Quality, the Office of Planning and Analysis, and the Bureau of the Budget. It included an exhaustive survey of the state-of-the-art in geographic information systems. This survey covered over 60 organizations in six countries. Today, phase I of IRIS exists and contains NARIS as a subset.

IRIS/NARIS is a parcel system*. It contains data classes which describe attributes of land parcels. The data structure supported (a two level heirarchy) is sufficiently flexible to accommodate natural resource, demographic, and economic data. The 18 classes of data currently stored fall into eight major groups: geology, land use, woody vegetation, soil, water, natural resources, employment, and population.

The original goal of NARIS was to provide a single agency with a tool to help analyze a regional, natural resource database. NARIS had to meet severe cost and performance requirements if it was to become an operational tool in day-to-day use. IRIS had more ambitious goals. It was to accommodate a much larger and more diverse data base. We hoped that IRIS would be of use, statewide, to all agencies, both public and private, involved in the planning process. In spite of the facts that the need for the system was perceived by the ultimate users, that the users played the major role in procuring funds, that the system had both high and low level support in government, and that many short and long term needs were correctly predicted by the computer system design team and adequately implemented, IRIS/NARIS cannot yet be called a success.

---

* In the IRIS Feasibility Study [1], we classified geographic information systems as area, point, or network systems. Area systems were further subdivided into uniform grid, parcel, and polygon overlay systems.

The modest goals of NARIS have been met. IRIS is now in day-to-day use in the largest planning agency in Illinois. That agency has planning authority for two thirds of the population of Illinois. However, the growth of the system to areas and agencies outside of northeast Illinois has not yet happened.

This history is important because it helps to illustrate the range of interests involved when building a system like IRIS/NARIS. The complexities involved in dealing with multiple agencies, with varying levels of skill and often conflicting approaches to planning cannot be eliminated - in fact, these complexities are the problem.

We learned and reinforced several concepts by building a large system of our own and surveying other systems.

1) The user of a user oriented system doesn't know what to ask for. He doesn't have the computer experience to know what capabilities computer technology will support and to predict system growth areas. The system designer must be exceptionally sensitive to the user and learn enough of the user's job that he can interpret user requests, suggest alternative directions to take, and predict future needs.

2) There is no such thing as "the" retrieval/analysis language. Each class of users has developed a jargon and a thought process tuned to his problem area. For each user there are aspects of his problem approach that are implicit. These are in conflict with other users who would like the same aspect of problem definition and solution to be explicit. We believe that the acceptability of the system and the productivity of the user are both significantly enhanced by tuning the retrieval and analysis language to the particular problem.

3) A geographic information system should store data which is objectively measurable on the ground. It should be able to construct interpretations from such "raw" data. In general, interpretations change in time and differ with analysis techniques. Data collection is the major cost of a geographic information system (often ten times more than the cost of the software [1]). Storing raw data is a necessary hedge against data obsolescence.

4) Parcel systems are superior to uniform grid systems. They are cheaper because they allow the use of previously collected data on census tracts, ownership parcels, etc. They can be made to conform to the geographic shape of data collection units (both naturally and politically drawn). Hence, they allow a more accurate model of the data. They can be made to conform to the data cells the planner already uses. Many system designers choose a uniform grid because it makes the data easier to address and the computer system easier to build. Unfortunately, the use of uniform grids usually requires a new data collection effort and a retraining process for the user. The cost and schedule implications of data collection and user re-education dramatically overshadow the small advantage a convenient grid choice provides for the computer system implementor.

5) Geographic information systems are very sensitive to costs. In order to generate systems of the necessary cost effectiveness, they must be carefully tuned to the data structures and analyses desired. Furthermore, the designers must plan for the rapid expansion of facilities and exploit data management software techniques which provide the maximum storage and processor efficiency.

2. The Problem

2.1 Data

The data base which supports the planner is very large. Once a computer system is available to manage it, the data base tends to expand to cover additional geographic areas and new data classes. However, data already entered changes slowly. During the IRIS feasibility study we found machine readable data that would occupy more than 800 million bytes when compressed to a minimum number of bits.

NARIS was designed to work with only one resolution of data and one addressing scheme. That was the 40 acre 1/4 1/4 section based on the public land survey. The public land survey (created in 1785) was supposed to produce a uniform grid. Unfortunately, surveying errors and other problems have produced some 1/4 1/4 sections have more or less than four sides and 40 acres. These idiosyncracies require a parcel system to handle the data and a rather strange addressing scheme in the retrieval language to mirror the parcel descriptions in current use.

As the IRIS program took shape, it became obvious that we had problems trying to expand NARIS. There was a need for several parcel resolutions. Some data is collected and used at a finer or more coarse resolution than other data. We had planners who needed a county data base, one based on census divisions, another on municipalities, or crop reporting districts, and even 160 acre 1/4 sections. Of course, each of these had a different addressing scheme. Sometimes these addressing schemes are incompatible. That is, they can't simultaneously exist in the retrieval language because of inherent ambiguities or because of human engineering considerations (the IRIS/NARIS solution is to allow only one parcel addressing scheme be active at a time).

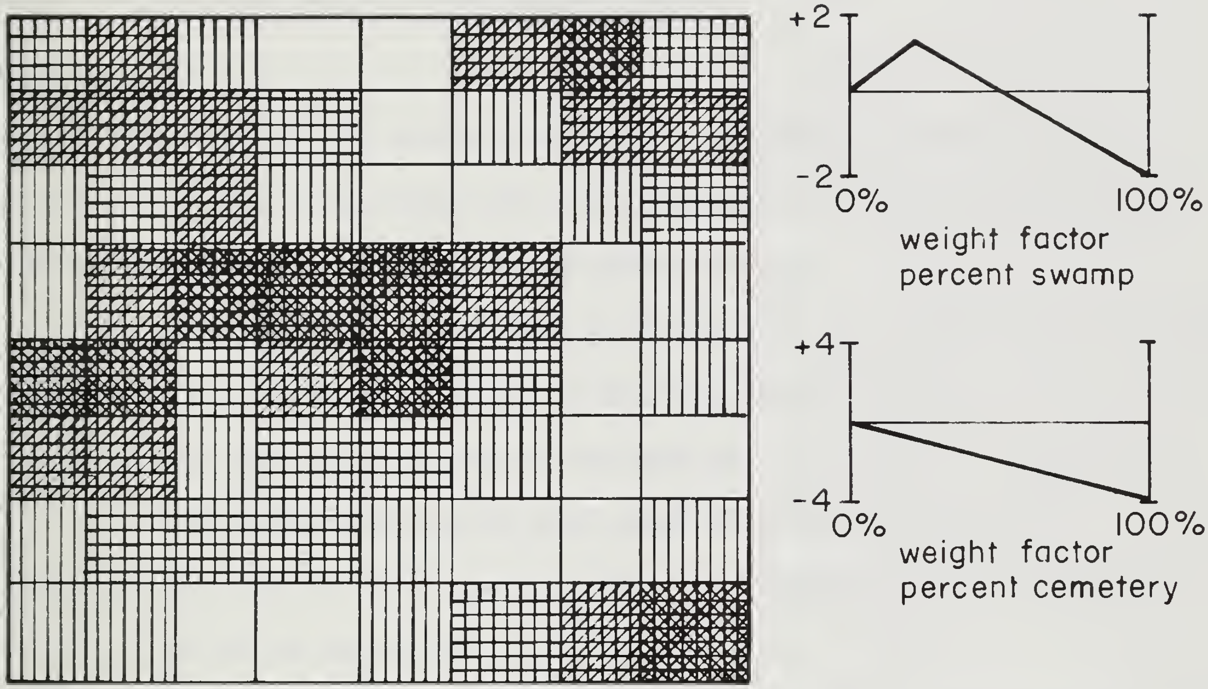
2.2 Analysis Techniques

A variety of analysis techniques were required by the users. These varied from the generation of simple tables of data about a single parcel, to summaries of the data over a set of parcels, to more sophisticated arithmetic analyses and map generation.

Over all these requirements there was one common theme: the need for an interactive system. The planner deals with enormous quantities of data and subtle relationships in that data. He had to be able to browse through the data to discover those relationships and explore alternative planning strategies.

The requirement for mapping and weighting function analyses is particularly interesting. The idea of a weighting function is to compute a parcel-by-parcel suitability for a particular land use. This is done by weighting each important attribute in the parcel positively or negatively depending on how the attribute mitigates for or against the proposed land use. All of the weights in the parcel are then summed to produce a composite suitability index. Some users of IRIS/NARIS had previously done manual weighting function analyses as large as 50 attributes over 18,000 parcels.

After the weights are calculated for each cell they are normally displayed as a map. Gray scales or differently colored regions of the map indicate varying suitability (see fig. 1). IRIS/NARIS produces gray scales on an electrostatic plotter. Color, pen plotter maps are currently available on the production system. Unfortunately, the plotter is 150 miles from the major user. As a result, pen plotter maps are not used. The most frequently used graphic is a terminal map with weights printed in each parcel. The user hand colors the parcels according to the printed weight.



Example of a weighting function analysis in a highway corridor study. A cell is negatively weighted if it contains a high percentage of swampland or cemetery.

Figure 1: Weighting Function Analysis

It is not always possible or desirable to incorporate existing models and analysis packages in the geographic information system. It is important that the user be able to use the data and analysis features which are in IRIS/NARIS to prepare machine readable files which can be input to other systems. This feature is exercised daily by IRIS/NARIS users.

2.3 Users

The users of the system have diverse capabilities and interests. Some are data collectors, others are urban planners or analysis specialists, and still others are administrators and policy makers.

A common profile of the user we should support, however, can be generated:

1. The user knows a subset of the data base very well.
2. The user wants access to raw data and the ability to build his own interpretations of data about which he is expert.
3. The user wants to use standard interpretations of data that he is not an expert on.
4. The user is not a computer scientist and has probably never used an interactive terminal.
5. The user is suspicious of computer scientists.

Perhaps the greatest problem with a large class of the users is that they are not accustomed to planning quantitatively. Before the advent of the computerized geographic information system, it simply was impossible to quantitatively analyze the enormous data base. Therefore, many planners got used to qualitative analyses and "flying by the seat of their pants". In order to get the system to this kind of planner, we have an unavoidable education problem. The planner will have to be taught how to quantize his opinions (e.g. how much is a farm worth relative to a small pond?) Fortunately, the weighting function

has come into acceptance as a quantitative planning tool at many of the larger planning agencies. This should help in the education process.

3. The Logical Structure of the Data Base

Figure 2 shows the IRIS/NARIS data base structure. The atomic data units are called data elements. All data elements are aggregated into data classes. In the figure, *forestry* is a data class with data elements *type*, *acres*, and *density*. Data element values may either be numerical quantities or character codes. In the forestry example, values for *forestry type* are the character codes *pine*, *oak*, etc. The values for *forestry acres* are numerical quantities ranging from 00.1 to 40.0. In the event that the value of an element is not recorded, a "missing data" value is stored.

A single class entry in the data base is called an occurrence. An occurrence has one set of data element values for each data element in the data class. There can be more than one occurrence for any data class in each parcel. Occurrences usually describe things like a single stand of trees or a single plot of soil in the parcel. In figure 2, parcel 1 has 3 tree stands. Two are pine and one is oak. There are a total of 34.1 acres of woody vegetation on the parcel.

Each parcel can have many data classes. However, all data must conform to the two-level, class-element structure with multiple class occurrences. This logical structure has been adequate for handling all the parcel attributes we have encountered.

The parcels in the data base are referenced directly by a parcel specification. This specification is tuned to the particular user and geographic

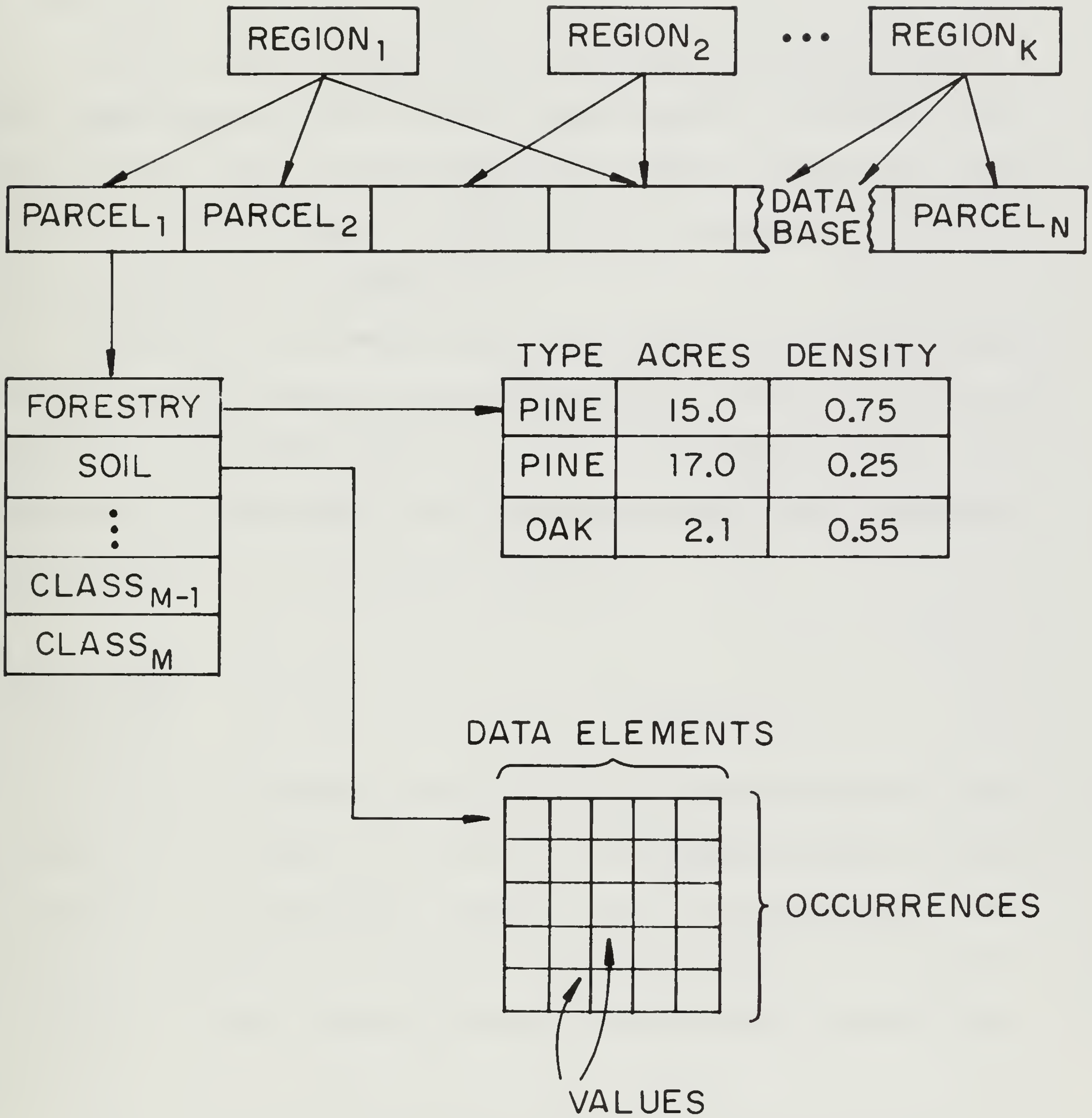


Figure 2: The Iris/Naris Data Structure

addressing scheme involved. The two schemes currently used are based on the public land survey and intended for use on the 1/4 section and 1/4 1/4 section data bases.

In normal use, individual parcels are seldom directly referenced. Instead, region names are used to refer to sets of parcels. Each parcel has an internal parcel number. Regions are indexes (inverted lists) which contain these internal parcel numbers in a compressed form. The parcels in a region need not be geographically contiguous.

In addition to the parcel data base and region files, there are many auxilliary files which provide on-line documentation. Every class, data element and non-numeric data element value can be listed and described on-line. Also, the request used to create any of the existing utilities (regions, functions, and abbreviations) can be recalled and displayed.

4. User Language

4.1 Introduction

The user language was intended to provide powerful retrieval capabilities in an interactive environment. The syntax of the language was made as English-like as possible to facilitate fast learning, ease of use, and clarity. All data classes and elements are referred to by full length names. All requests have a simple, consistent sentence structure. Requests may be several lines long.

There are two basic categories of requests: those that retrieve data from the geographic data base and those which perform auxiliary actions. A retrieval request is composed of phrases that specify the data elements to be accessed, the action to be executed on the data, the scope to be acted upon

(a region), and the destination of the results of the retrieval. Each action has a unique verb in the request language: *tabulate*, *calculate*, *map*, and *output*. *Tabulate* lists information on a parcel-by-parcel basis. *Calculate* produces summaries of data over the complete region. *Map* produces scaled and accurate maps of parcels that are gray level shaded according to the computed value of an arithmetic expression. *Output* produces a disk file containing the value of one or more arithmetic expressions for each parcel. The *output* verb allows other programs such as report generators, statistical systems, simulators, and site location algorithms to manipulate the data outside of IRIS/NARIS [2,3].

Regions allow a user to manipulate data using a geographic notion that is ingrained in their thought processes. By limiting each request to the parcels specified in a region, serial searching of the entire data base is eliminated.

Phrases that specify the data elements to be retrieved can be as simple as a single data element name or as complex as an expression containing functions and Boolean and arithmetic operators. Parcel weighting schemes are specified by arithmetic expressions. At the lowest level, these expressions add, subtract, multiply and divide values of numerical data elements within the same occurrence. By combining low level arithmetic expressions with Boolean expressions, a single value for a class can be computed on a subset of the occurrences in a parcel. A single value or weight for a parcel can then be computed by arithmetically combining the values calculated for different classes.

Auxiliary requests permit users to create and maintain various utilities or textual descriptions useful in the construction and execution of retrieval requests. For example, an abbreviation utility permits a user to abbreviate portions of a request with a single name. This gives the language a text replacement

macro capability which reduces verbosity for the familiar user, permits long requests to be broken into smaller segments, and lets nonsophisticated data users easily execute requests created by technical specialists. English text descriptions of data classes, elements, and values are available through the *what is* request. The *what is* request also allows users to recall the definition of a utility (a region or abbreviation function). Other facilities in the language are used to manage the various utilities. These facilities allow a user to save a utility, destroy a utility, or control which users and agencies have access to a utility.

The following sections will give a more precise description of the user language. Examples of requests and phrases will be used in conjunction with a flow diagram of the syntax. Data class and element names as well as data values will be drawn from data bases currently in the IRIS/NARIS system. Detailed descriptions of the data element values used in examples will not be given. The user normally obtains this information directly from the system. For the purposes of this paper, only a general description of the request is necessary to understand what the system is doing.

4.2 Data Expressions

Data expressions denote data elements to be accessed, specify arithmetic calculations to be performed on values retrieved, and specify which data occurrences should be included in higher order calculations. Each reference to a data name in an IRIS/NARIS request implies the retrieval of the corresponding data values from the data base.

4.2.1 Simple Expressions

A simple expression is a data element name. Some of the data element names in the *soil* class are *soil number*, *soil acres*, and *soil erosion*.

4.2.2 Arithmetic Expressions within a Class

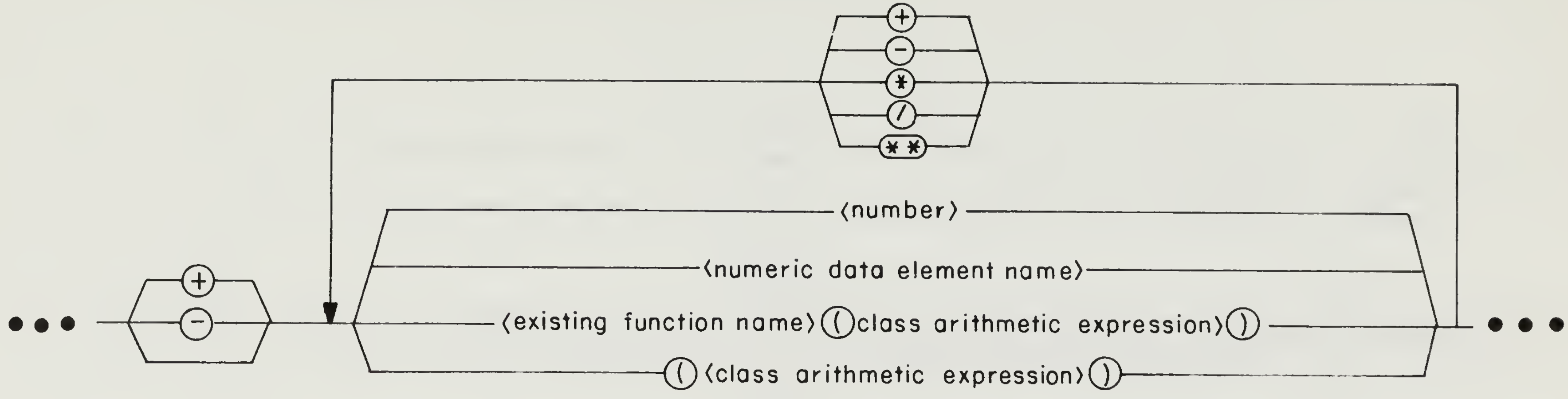
A class arithmetic expression specifies the retrieval and calculation of values within a single class occurrence. Figure 3 shows the syntax of this phrase along with an example. The arithmetic operations supported are unary supported are unary plus and minus (+,-), addition (+), subtraction (-), multiplication (*), division (/), and exponentiation (**). All arithmetic is real. Functions, somewhat like Algol function calls with one argument, can also be used in arithmetic expressions. The traditional precedence of operators is used and can be altered, using parentheses, in the usual manner.

4.2.3 Boolean Expressions within a Class

A Boolean facility is provided to specify which occurrences qualify for higher order calculations or data retrievals. This facility, called a class Boolean expression, returns a true, false, or maybe value for each occurrence in a single class. Figure 4 shows the syntax for class Boolean expressions. A class Boolean expression is given the value true if the relation is satisfied for the arithmetic expressions or data elements involved. If the relation is not satisfied then the value is false. If the relation is indeterminable, because of missing data or execution errors (e.g. divide by zero), then the value is maybe. The logical operators *and* and *or* with traditional precedence and expression parentheses are supported. Figure 5 shows the true, false, maybe logic for *and* and *or*.

4.2.4 Arithmetic Expression over a Parcel

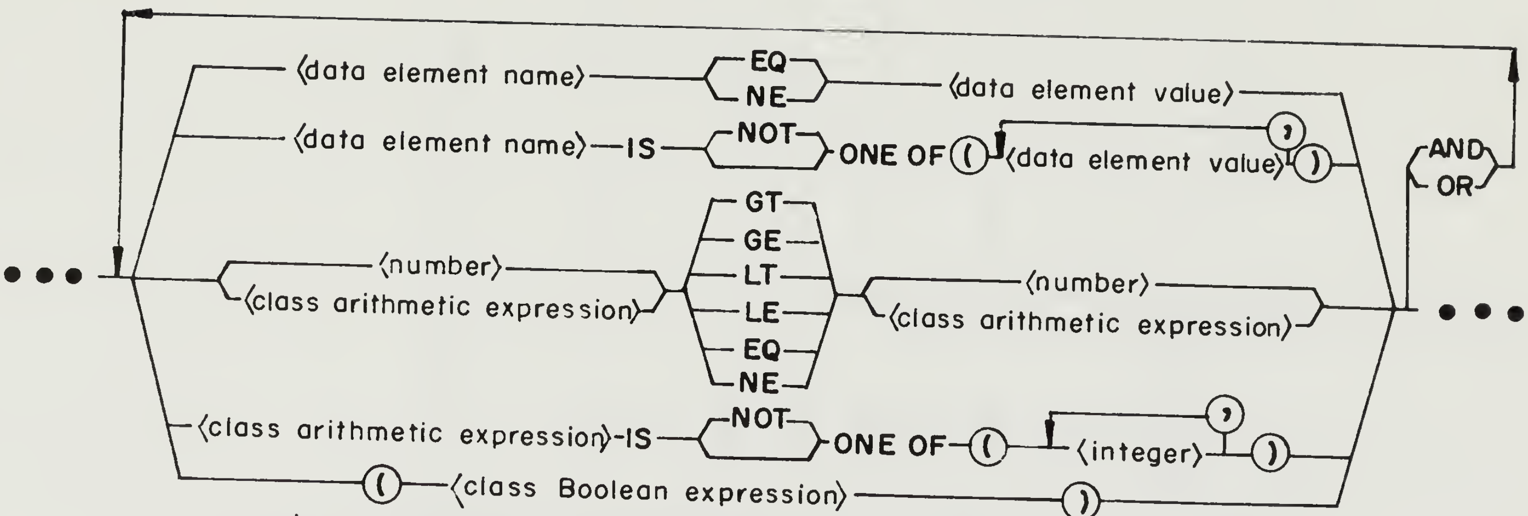
Parcel arithmetic expressions (PAE) produce a single arithmetic value for each parcel. Figure 6 shows the PAE syntax. The rules for using PAE primaries to form PAE's are exactly analogous to the rules for class arithmetic expressions.



example : **FORESTRY ACRES + (ECOLOGYWTFCN (ACRES * DENSITY) ** .5)**

notes :
1. a class arithmetic expression must contain at least one numeric data element name.
2. all data element names must be in the same class.

Figure 3 : Class Arithmetic Expression



examples: **SOIL NUMBER EQ W103 AND SOIL ACRES GT 10**
SOIL ACRES*4 GT 30
SOIL NUMBER IS ONE OF (210, W103,W)

note: all data element names must be in the same class.

Figure 4: Class Boolean Expression

OPERANDS		OPERATIONS	
b1	b2	b1 AND b2	b1 OR b2
true	true	true	true
true	false	false	true
true	maybe	maybe	true
false	true	false	true
false	false	false	false
false	maybe	false	maybe
maybe	true	maybe	true
maybe	false	false	maybe
maybe	maybe	maybe	maybe

Figure 5: True, False, Maybe Logic for AND and OR

A PAE primary is shown on the top line in figure 6. Function names may also be used in PAEs.

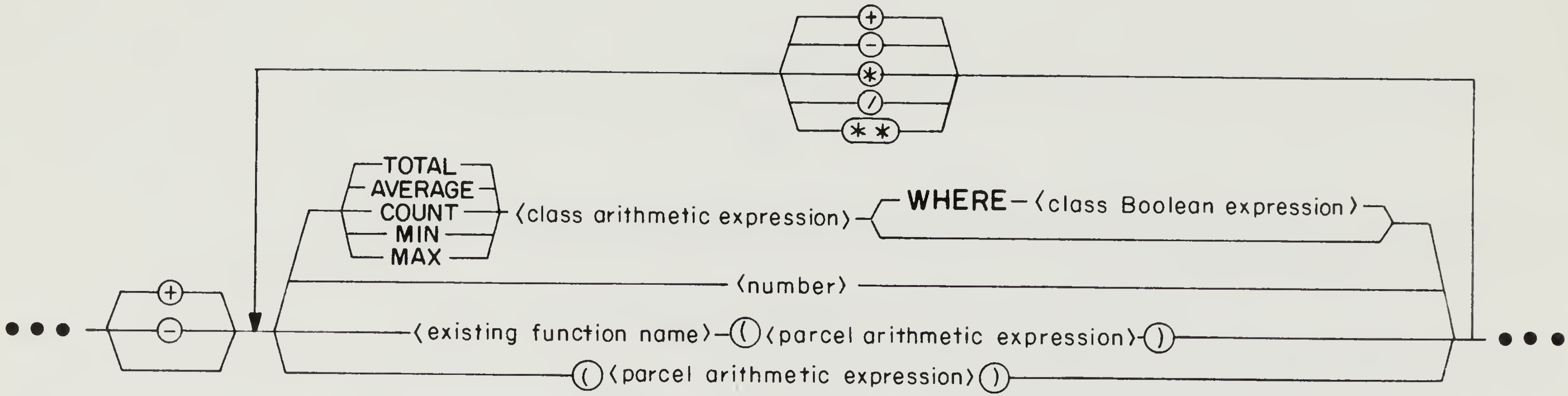
A PAE primary is a rule for computing a single result for one class in one parcel. A vector of values is produced by the evaluation of the class arithmetic expression on each qualifying data occurrence within each parcel. If a class Boolean expression is not used then every occurrence within the class is a qualifying occurrence. When a Boolean expression is included in the PAE primary, the expression is evaluated on each occurrence. If the resulting value is false then the data occurrence does not qualify. If the value is true then the occurrence qualifies. If the value is maybe, then the computation is terminated on the parcel and the user is notified.

The resulting vector of values are operated upon by the algorithm associated with the specified functional designator. *Total* causes the final result of the PAE primary to be summation of values in the vector. *Average* computes the average value. *Min* and *max* find the minimum or maximum value respectively. Count returns the number of values in the vector (equal to the number of qualifying occurrences).

A PAE primary may contain a class Boolean expression which disqualifies all of the occurrences. In this case, a value of zero is assumed for the result of the primary regardless of the functional designator. This condition is noted by the system and indicated to the user if necessary.

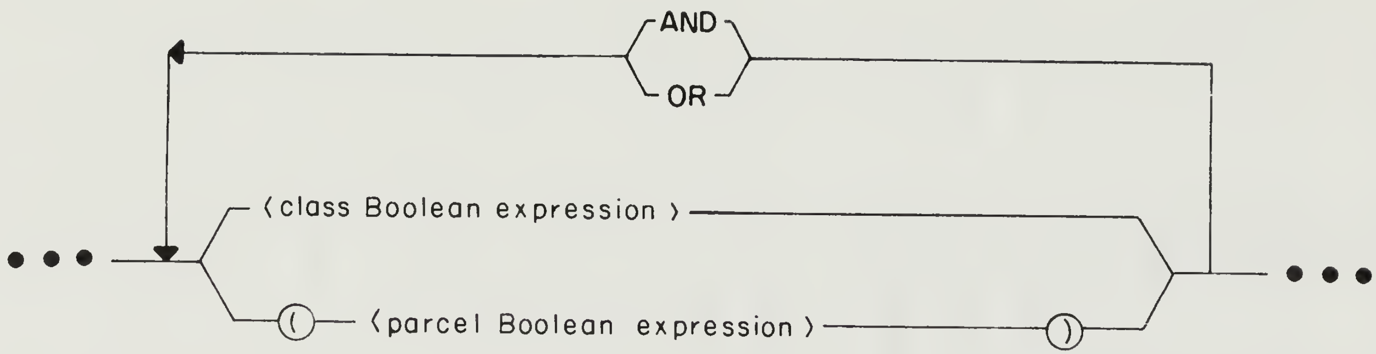
4.2.5 Boolean Expressions over a Parcel

Parcel Boolean expressions are used to produce a single true, false, or maybe value on all the data classes and their occurrences within a single parcel. The syntax of parcel Boolean expressions is given in figure 7. The primary for a parcel Boolean expression is a class Boolean expression. The



example: TOTAL SOIL ACRES WHERE SOIL NUMBER EQ 103 + AVERAGE FORESTRY DENSITY

Figure 6: Parcel Arithmetic Expression



example: SOIL NUMBER EQ W103 AND FORESTRY ACRES * DENSITY GT 10

Figure 7: Parcel Boolean Expression

rules for the evaluation and precedence of operations are analogous to class Boolean expressions.

4.3 Creating Regions, Functions, and Abbreviations

4.3.1 Regions

Regions are one of the more important concepts in the IRIS/NARIS system. The region request allows a user to select, in language familiar to him, a portion of the data base for further analysis. This eliminates unnecessary serial processing. It reduces the cost of using the system, the time needed to answer a request, and the amount of output that a user must wade through.

Regions point to subsets of the parcels in the data base. Figures 8 and 9 present the syntax for creating regions. They may be created in four different ways:

(1) by explicit geographic description using standard Public Land Survey notation to address one or many parcels;

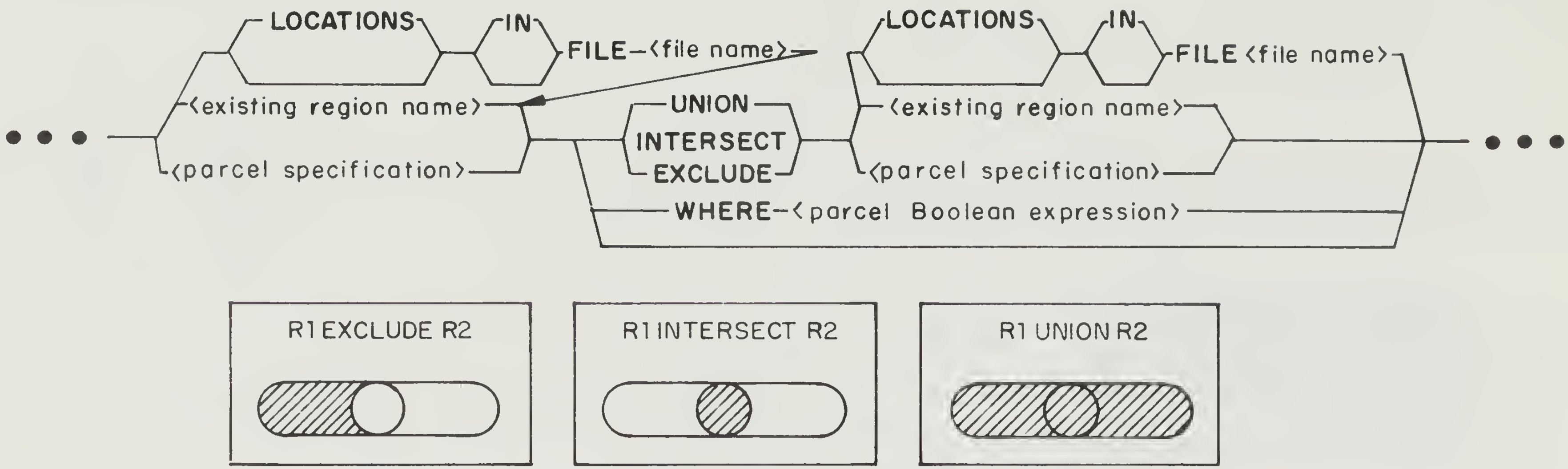
(2) from the data content of parcels using Boolean expressions to specify the conditions that data values must satisfy;

(3) from the geographic locators attached to external data files;

(4) by constructing new regions from existing regions using the set operations union, intersect, and exclude.

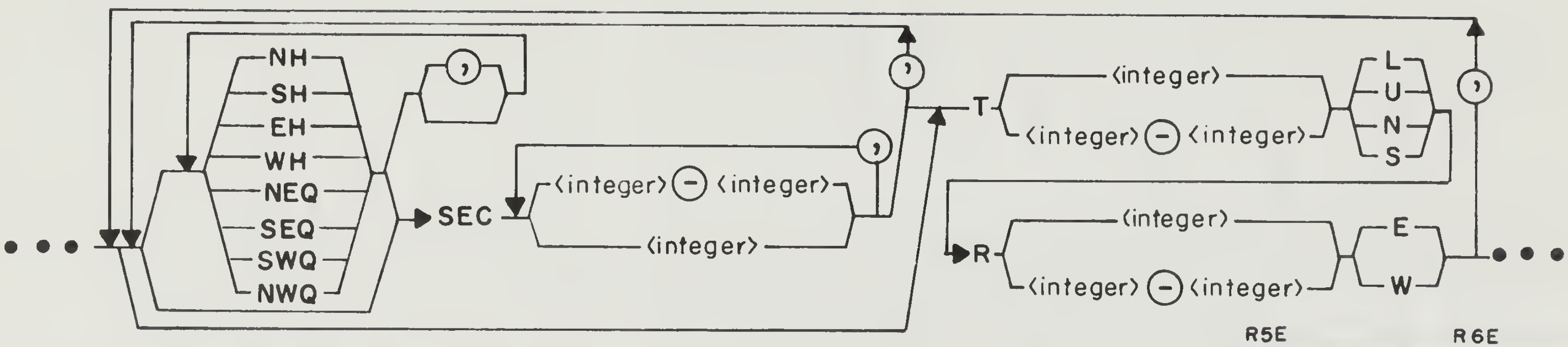
Regions, internally, are inverted list constructs. However, since they are completely user specified, there is no system guessing as to what should be indexed or how limited the data analysis can be to form an index. This design is unique to IRIS/NARIS but can be useful to other data systems maintaining slowly changing data bases.

REGION <new region name> IS <region expression> #



example: **REGION SUBURBS IS COOK EXCLUDE CHICAGO #**

Figure 8: Region Request and Region Expression



example: **T44N R63, SEC 1-3, 10-15, EH, NEQ SEC 4 T44N R5E**

note: this is public land survey notation.

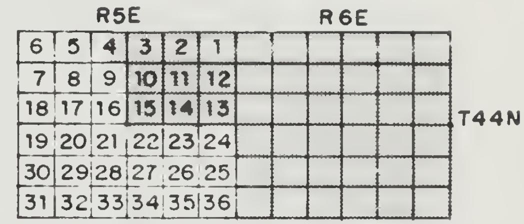


Figure 9: Parcel Specification

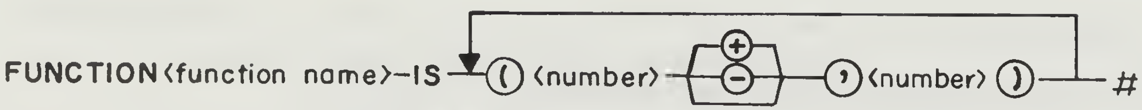
4.3.2 Functions

A *function* request is used to define a function to IRIS/NARIS. The syntax of this request is found in figure 10. In an arithmetic expression (figures 3 and 6) the argument for the function is enclosed in parentheses following the function name. A function is defined by specifying the x,y coordinates of the end-points of the line segments of a piecewise linear function of one variable. This provides an easy to use heuristic numerical transformation capability. The defining coordinate pairs are listed by giving the argument value (x-coordinate) first and the function value (y-coordinate = f(x)) second. Coordinate pairs must be listed so that a function is unambiguously defined: that is, by specifying the x-coordinates in increasing order from left to right.

The limit indicator is used to specify discontinuous functions. The "+" sign after the coordinate "x" in the coordinate pair (x+,y) states that the function value "y" is not given to the argument value "x" but to the next representable argument value larger than "x". Similarly, the x-coordinate "x-" indicates a number just less than "x".

4.3.3 Abbreviations

Requests contain frequently used, sometimes lengthy, names and phrases. Each user needs to have his own abbreviations for those phrases he uses. There is also a need to allow nonsophisticated data users to easily execute requests or phrases (data interpretations) created by technical specialists. An *abbreviation* request allows a user to specify a single word that represents an arbitrarily long text string. Figure 11 shows the abbreviation syntax. An abbreviation name can appear in any request and must be followed by a period. The system expands abbreviations as it compiles a request. The depth of nesting is a system parameter.



example: **FUNCTION F IS** (−3,−2) (−1.5,0) (1−,1) (1,0) (6,3)#

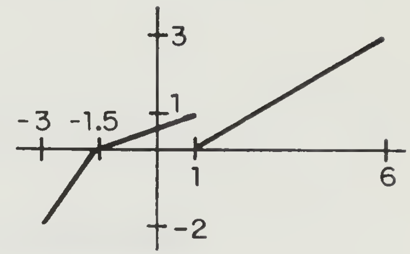


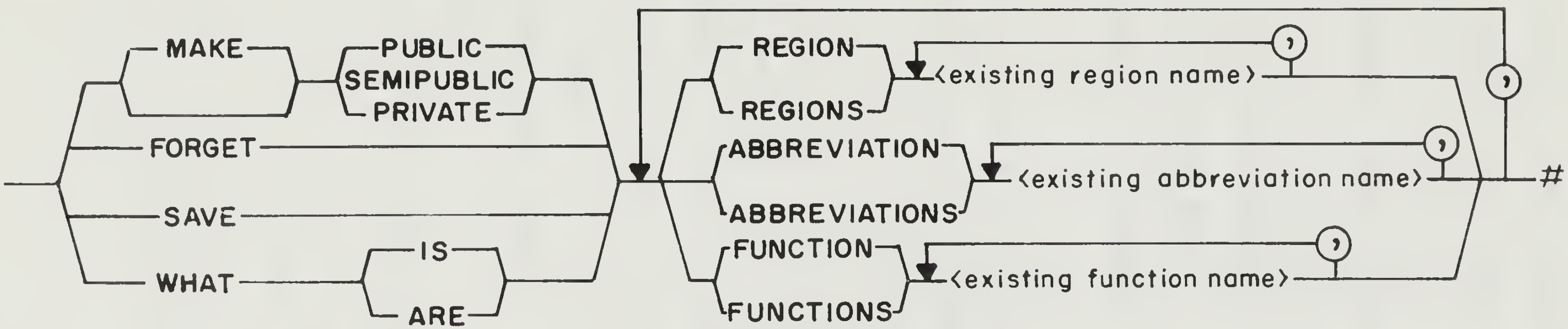
Figure 10: Function Request

ABBREVIATION-<new abbrevation name>-IS-any sequence of characters#

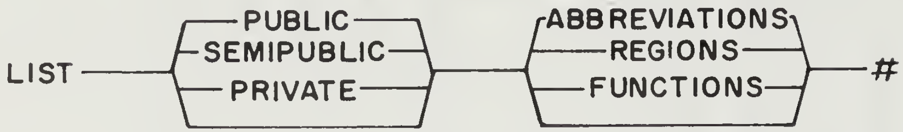
example: ABBREVIATION SILTLOAMSOILS IS
24,59,91,145,146,147,148,149,155,189,192,193,206,221,223,224,228
229,231,241#

example of use: TABULATE SOIL WHERE NUMBER IS ONE OF (SILTLOAMSOILS.)#

Figure II: Abbreviation Request



example: FORGET ABBREVIATION A1., REGIONS ONE, TWO #



example: LIST PRIVATE ABBREVATIONS #

Figure 12: Utility Management Requests

4.3.4 Utility Information and Housekeeping

Regions, functions, and abbreviations can be stored at three different access levels: private, semipublic, and public. The syntax to save, forget, change, or display status, retrieve definitions, or list the names of utilities is shown in figure 12. Each user has the only access to his private utilities. Several users can be collected together to form a semipublic access group such as one organization or planning agency. Utilities at this level are available to all users in the appropriate semipublic group. Public utilities are available to all users.

Utilities can only be created at the private level. A region, function, or abbreviation request is used. In order to save this information for use at another session, a user must execute a *save* request. To retrieve the names of the utilities stored, a user can type a *list* request. In order to retrieve the statement that created the region, function, or abbreviation a user can issue a *what is* request. To destroy a utility, using a *forget* request, a user must have been the one who created the utility originally and it must currently exist at his private level. The access level of a utility can be increased or decreased by issuing a *make* request.

4.4 Retrieval Requests

The user can request several different types of retrievals and reports based on the parcels in a region. Four retrieval verbs are provided: *tabulate, calculate, output,* and *map*. A syntax of retrieval requests appears in figure 13. Retrieval requests are built up out of all the phrases that have been previously discussed and an additional phrase to specify the destination of the final report, file or map.

Parcels whose data values result in maybe conditions or execution errors are not included in the final report of a retrieval request. Instead, a special region is created called the "error region". It exists until written over with a new error region. This region contains all the parcels causing trouble. The parcels in the error region can then be accessed to determine the data occurrences causing execution errors. This facility allows the system to keep control when execution errors occur, without losing the information the user needs to analyze what happened.

4.4.1 Tabulate

Tabulate generates automatically formatted reports on high speed line printers or at the user terminal. It lists only the specified data on a parcel-by-parcel and occurrence-by-occurrence basis.

4.4.2 Calculate

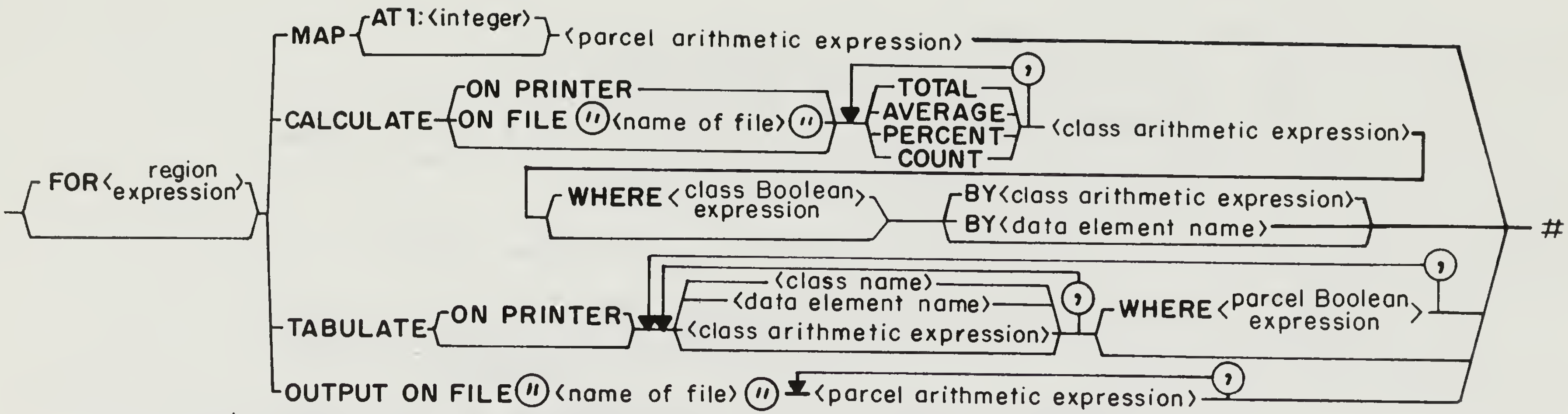
Calculate produces summaries on data within a region and ignores parcel boundaries. The summaries can be grouped and sorted by arithmetic value or by data element value. Reports can be generated on a printer, disk file, or at the user terminal.

4.4.3 Output

The *output* request creates a disk file of numerical calculations on a parcel-by-parcel basis. Each record in the output file has a geographic locator derived from the data in the *survey* class in each parcel. Along with each geographic locator are stored the values of each parcel arithmetic expression.

4.4.4 Map

The *map* request can produce scaled and accurate plots of regions on five different devices. The data that determines the points of the polygon describing the parcel are stored in a data class *mapdata*. Each parcel in the



examples: FOR T44N R5E MAP AT 1:40000 TOTAL FORESTRY ACRES#
CALCULATE ON PRINTER AVERAGE TOTAL SOIL ACRES BY NUMBER WHERE EROSION IS 3 #
TABULATE SOIL FORESTRY COVERTYPE#
OUTPUT ON FILE "MY FILE" AVERAGE SOIL ACRES WHERE NUMBER IS 103#

Figure 13: Retrieval Requests

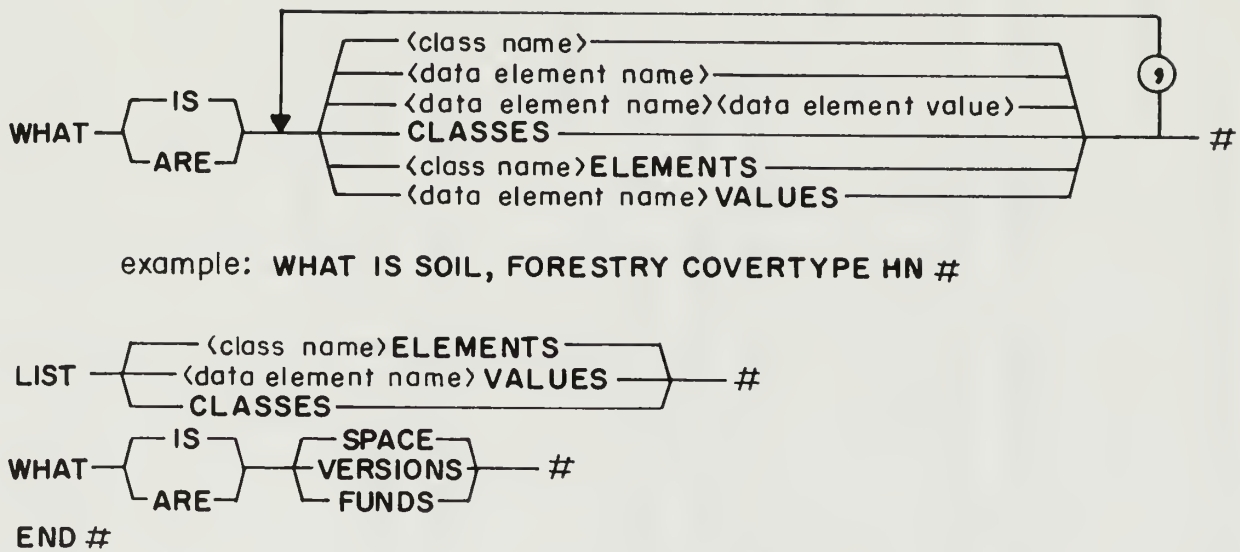


Figure 14: Data Description and Miscellaneous Requests

region is shaded. The gray scale is proportional to the value calculated by the parcel arithmetic expression. Maps of regions whose parcels approximate a regular grid can be produced on a printer or at the user's terminal.

4.5 Data Description and Miscellaneous Requests

Figure 14 shows the syntax used to retrieve data names, data descriptions and system usage statistics.

4.5.1 Data Names

The *list* request was partially described in section 4.3.4. It can also be used to retrieve the names of the current classes and the names and allowed values of the data elements in each class.

4.5.2 Data Descriptions

Textual descriptions of all data classes, elements and non-numeric values are stored in the system. These are most useful to users who may not know the meaning of some names and codings. This information is accessable through the *what is* request.

4.5.3 System Statistics

A user can also request system statistics using the *what is* request. The user's disk space, remaining funds, and the current version number under which IRIS/NARIS is running are accessible.

5. System Architecture

5.1 Introduction

IRIS/NARIS is coded in Algol for a Burroughs B6700. The first system was brought up on our own machine at the Center for Advanced Computation. It was transferred to a more service oriented and cheaper B6700 at the University

of California at San Diego. The system is currently used and maintained from Illinois over the ARPA Network.

Until recently, IRIS/NARIS ran as a separate Message Control System (MCS) on top of the B6700 Master Control Program (MCP). It was easier and more efficient to support multiple users and provide the appropriate file interlocks as an MCS. However, because an MCS communicates closely with the MCP, it requires constant maintainance to keep up with MCP modifications. Billing and privacy problems arose as more users exploited IRIS/NARIS and external statistical [2] and optimal site location [3] packages. As a result, the system has been changed to run as a separate program under each user's account, and is no longer an MCS. This has greatly simplified the maintainance and billing problems at the cost of a less efficient interlock scheme for some shared, writable files.

The single resolution NARIS was designed over a three month period and coded by four programmers over an 18 month period. The first production NARIS was approximately 20,000 lines of heavily documented code. The current IRIS/NARIS code is 30,000 lines and is more efficient and powerful.

A lot of attention was paid to modularizing the system so that even basic tables could be altered without requiring system-wide modifications. Access to any table is only permitted through a small number of procedures or macros. Other program statements are not allowed (by programmer agreement) to directly access those tables. As a result, critical tables, used throughout the system, can be (and have been) radically altered without difficulty.

By setting compile-time options, debugging and tuning versions of the system can be generated. The trace, dumps, and timing features are particularly extensive. The time spent in any procedure, the procedures it calls and the

total can be recorded for each procedure and user request. The emphasis on instrumentation has paid off in system efficiency.

Figure 15 shows the basic system architecture. In addition to the files and programs shown, there are back up facilities and tape rotation schemes to protect these files from system failures. In addition, various log and status files allow full recovery from system crashes.

5.2 Data Definition and Insertion

The left side of figure 15 is concerned with creating, correcting, and enlarging the data base and other files used at retrieval and analysis time. This is a batch activity.

There are three major disk files: the data base file, the symbol table, and the parcel dictionary. Each of these files has a header record which contains version information and other file attributes. The header contains the version number of the file and the version numbers of the other files and programs which are required for correct operation. This provides a run time consistency check of the system and its files. It prevents operator, system programmer, or data administrator errors from turning into major catastrophies.

There are four main programs associated with data definition and insertion: the parcel generator, the symbol table generator, the preprocessor and the data base inserter. Each runs as a separate batch program.

5.2.1 Data Base

The data base file is the heart of the system. The first record is the header record. All other records contain parcel data - one parcel per record. If the parcel data overflows a record, an overflow record is dynamically created and a pointer to the overflow record is planted in the previous record. An arbitrarily large number of overflow records can be used for any parcel.

At the head of each record is the overflow pointer to the next record and a count of how many bytes of class data exist in the record (see figure 16). At the end of the first record are the class pointers. Overflow records do not have class pointers. Each class pointer has two fields. The first is the number of occurrences of that class. The second is a pointer to the first byte of the first occurrence of the class. This pointer may point to a byte which is many overflow records away.

Some classes will have only one occurrence containing identical data values for a large number of their parcels (e.g., forestry data frequently specifies zero trees). In this case a single default occurrence is established. It is stored once so that it does not have to be repeated in the data base. A special value in the number of occurrences field in the class pointer is used to flag this condition. The default occurrence for each class is recorded in the symbol table and is available in a core array.

The data for each class and each occurrence begins on a byte boundary. The data element fields may begin and end at any bit within the occurrence (see figure 16). The result is a very dense data structure that is still relatively easy to pick apart using the byte and bit operators available on most machines.

Record access times are fairly large compared to transfer time from the disk. Furthermore, user requests frequently access parcels whose records are near each other in the data base. The data base is blocked at around 20 records/ block to take advantage of this. If an average of 2 parcels are used per block read, there is a net reduction in access time and CPU time. The data base is periodically reblocked and the record size changed to improve storage and access efficiency.

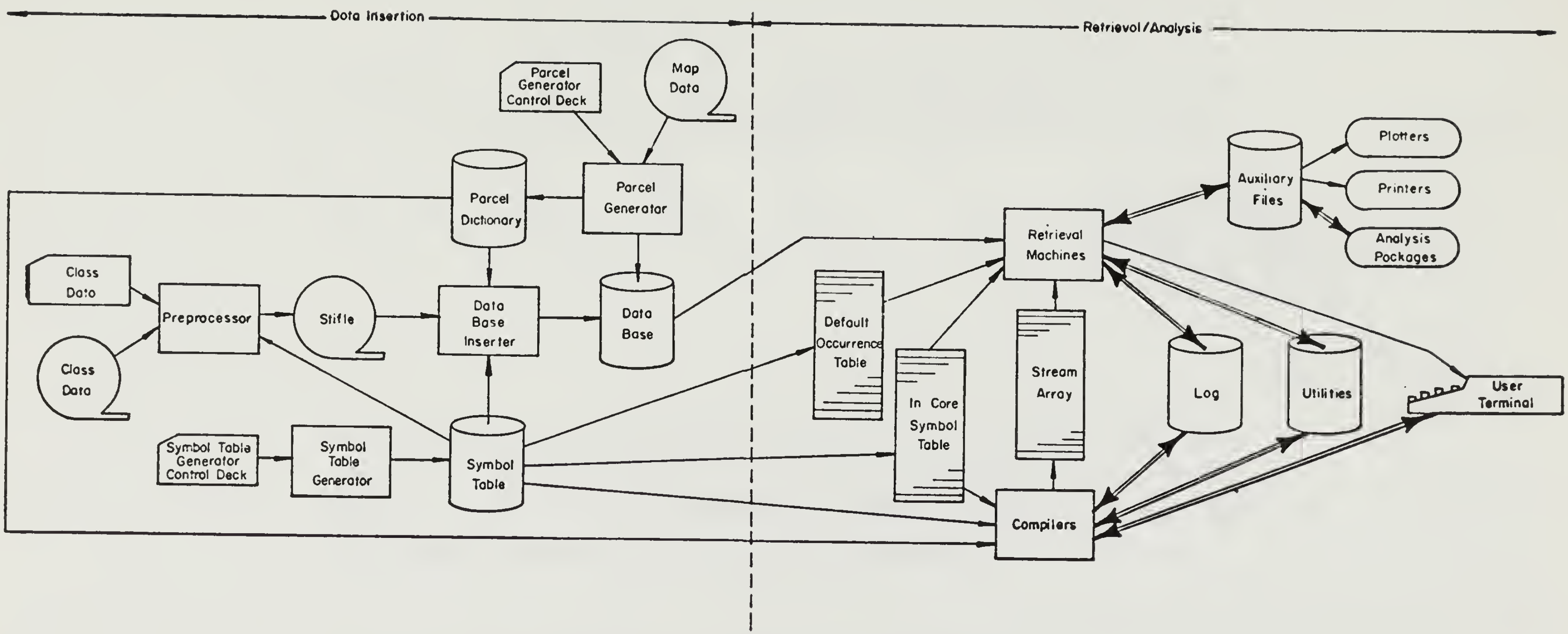
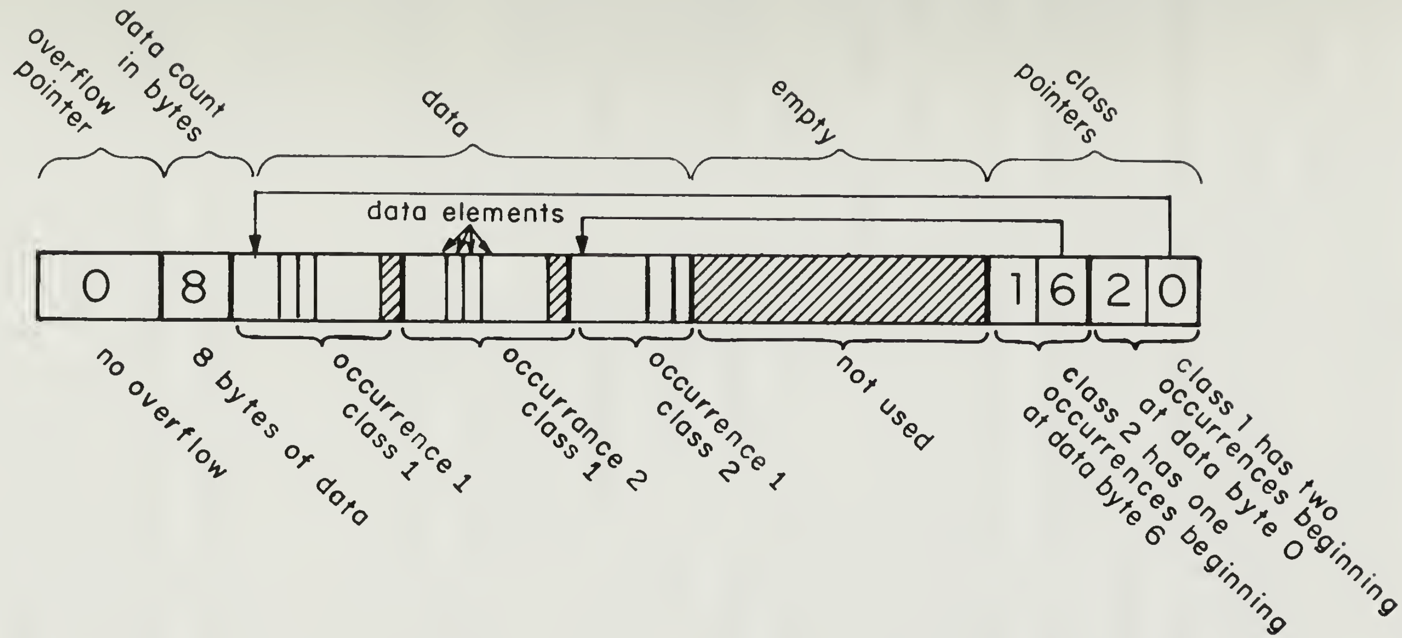


Figure 15: Information Flow in IRIS/NARIS



This record has no overflow, eight bytes of class data and two classes. Class one occuppies 3 bytes per occurrence but does not use all of the bits to record its 4 data elements. Class two fills its 2 byte occurrence with 3 data elements.

Figure 16: Data base record example

The combined effects of the compression, default occurrence and blocking techniques significantly reduce I/O time. As a result, the system is slightly CPU bound and response time is minimized.

5.2.2 Symbol Table

The symbol table contains complete information about the data base classes and data values. Specifically, it contains the name of each class and element, the length of each class occurrence, the position and size of each data element in a class occurrence, the values allowed for each data element, the position of each class pointer in the first record of a parcel, and a detailed textual description of each class, element, and value in the system (for use by the *what is* request).

Some of the data can have long character values. The symbol table contains hashing scheme parameters, if necessary, to help the compilers and data input programs rapidly locate the value. All data element values are stored internally in a minimum number of bits. The symbol table contains the compression and decompression parameters.

The most important identifiers, positional information, and compression parameters in the symbol table are copied into core at run time to reduce compile time.

5.2.3 Parcel Dictionary

The parcel dictionary is used to convert geographic parcel specifications into internal parcel numbers. An internal parcel number is the logical record number of the first data record for each parcel. The parcel dictionary is used during data insertion and also during retrieval.

5.2.4 Parcel Generator

The parcel generator adds new parcels to the data base. It also adds two initial classes to each parcel: a *mapdata* class which contains parcel boundary coordinates and a *survey* class which contains parcel identification. These data classes make each parcel self defining and eliminate the need for auxiliary description files at retrieval time. The *map* class is used by the *map* verb to prepare graphic output. The *survey* class is used by the *tabulate* and *output* verbs to annotate tables and construct geolocators for output files. The parcel generator reads a control deck for directions and scans a large file of digitized map coordinates prepared by the Illinois Geological Survey. The map coordinates on the tape covers the entire state of Illinois to the section level. Interpolation is used to produce 1/4 and 1/4 1/4 section map coordinates.

5.2.5 Symbol Table Generator

The symbol table generator reads a control deck to add new classes to the symbol table. The control deck contains information on classes, elements, and values including text information for the *what is* request. The type and range of each data element value is also specified. The symbol table generator uses this information to generate hash coding parameters, if necessary, and to automatically compute a minimum bit compression scheme and occurrence position for each data element. Consistency of the information is rigorously checked. No symbol table is generated if errors are found.

5.2.6 Preprocessor

The preprocessor and data base inserter programs work together to put data into the data base. The raw class data is read by the preprocessor, cleaned, compressed using the appropriate scheme and written out as a Standard Intermediate File (STIFLE). The STIFLE was to have provided archival backup

of the data base in the event of a catastrophic loss of the data base and its back-up files. Also, it was incorporated into the system to avoid data losses due to programmer errors. Otherwise, the programmers would be constantly modifying the data base inserter to handle various data idiosyncracies. In addition, the mnemonic format of the STIFLE was supposed to permit some radical restructuring of the data base classes without starting from scratch and a variety of old input media.

5.2.7 Data Base Inserter

The data base inserter reads the STIFLE file and inserts the data into the data base. In retrospect, the STIFLE has been valuable as a technique to separate the volatile and non-volatile aspects of data insertion. However, the data base backup system has proven to be very reliable as contrasted with the STIFLE tapes, many of which are no longer readable. The STIFLE tapes are useless as an archive and are no longer saved.

5.3 Retrieval and Analysis

Requests are processed in two phases. A compile phase is followed immediately by execution of a pseudo retrieval machine. Each major request and phrase in the user language has its own compiler and pseudo machine.

5.3.1 Compilers

The job of the compilers is to parse user requests and generate intermediate code for the pseudo machines. Intermediate code is placed into the stream array. If the compilation is not successful, then the appropriate error messages are sent directly from the compilers to the user.

There is a compiler for each IRIS/NARIS verb--*tabulate*, *calculate*, *output*, and *map*--as well as each major phrase -- arithmetic expressions, Boolean expressions, and parcel specifications. The verb compilers call the phrase

compilers which in turn can call other phrase compilers in a strict lattice pattern. Recursion is not used. Requests which do not require intermediate code, e.g., *what is, list,* etc. are executed directly in the compile phase.

Information in the symbol table and the parcel dictionary is used to recognize reserved words, data names, data element, value codes and geographic descriptions. Utility files are accessed to obtain abbreviations and functions.

5.3.2 Retrieval Machines

If the compilation is successful, control is passed to the retrieval machines. It is the job of each machine to interpret the intermediate code in the stream array and execute it by simulating a sophisticated stack machine. A machine exists for each compiler. The machine lattice structure is an exact image of the compiler lattice structure.

Because of the heavy use of Boolean interrogation of the data base, the Boolean machine provides a unique optimization feature. The Boolean compiler stores its intermediate code as a tree in the stream array. If one branch (operand) of a Boolean operator determines (predicts) the outcome of the operation (e.g. true for an *or* operation), the other branch is not evaluated. Thus, the speed with which a Boolean expression is evaluated is dependent upon which branches are taken first. The IRIS/NARIS Boolean machine keeps track of the predictive record of each branch in the expression and attempts to evaluate the most predictive branch first. During retrieval, the Boolean machine will continually switch the order of evaluation of the branches based on the cost of evaluating the branch and the recent history of data evaluations. As a result, the cost of expression evaluation rapidly flattens out and appears independent of expression complexity. The details of this technique are left for a future paper.

5.3.3 In Core Data Structures

There are three data structures that remain in core at all times because of their frequency of use: the incore symbol table, the stream array, and default occurrences. The incore symbol table contains four subtables:

(1) name table: contains all the names of data classes, data elements, and reserved words.

(2) value table: contains the values and compression parameters for each data element.

(3) data element information table: describes the position of data elements within class occurrences and points to relevant entries in the value table.

(4) class information table: describes the position of class pointers in the parcel record and points to entries in the data element information table and default occurrence table.

The stream array stores the compiled code used during retrieval. It is distinguished from most linear code streams in that the code is organized into blocks. These blocks are relatively independent of each other. They insulate one compiler from another. For example, *tabulate* requests may contain Boolean expressions. To compile them, the tabulate compiler calls the Boolean compiler which then compiles a block of Boolean code. During retrieval, the tabulate machine calls the Boolean machine to execute the block of Boolean code which the Boolean compiler generated. Thus, changes in the syntax of Boolean expressions or in Boolean machine pseudo operators only require changes to the Boolean compiler and machine. No modifications are needed in the tabulate compiler and machine.

5.3.4 Utility Files

Utility files are used to store all user-dependent names and the information associated with them. Each user can access three utility files: his private file, the semipublic file of his organization, and the public file. The files are hashed to eliminate most of the I/O time required to find a name on disk. The structure is shown in figure 17. Note that the names and data associated with all utility types (regions, abbreviation, and functions) are stored in a single file.

5.3.5 Log File

The log file has four kinds of entries: user, history, maintenance, and data base. User blocks record all of the attributes of each user, (e.g. account number, permissions, defaults, etc.). History blocks record the text of each request, the cost, the date and time, the amount of CPU, I/O, connect time, other resources consumed and certain notable execution conditions. Maintenance blocks log selected system error messages and unusual conditions. Data base blocks record the name, resolution and number of times each data base was used. Log analysis routines are available for billing, debugging, and tuning. The log contains precise information on load factors and human interaction that can be used to improve the retrieval algorithms, the retrieval language, and the diagnostic messages for user errors (there are over 400 error messages at this time).

5.3.6 Auxiliary files

Various auxiliary files comprise the last group of disk resident files. These files are used to store data produced by the *map*, *output*, and *calculate* verbs. The programs that draw maps also access the auxiliary files.

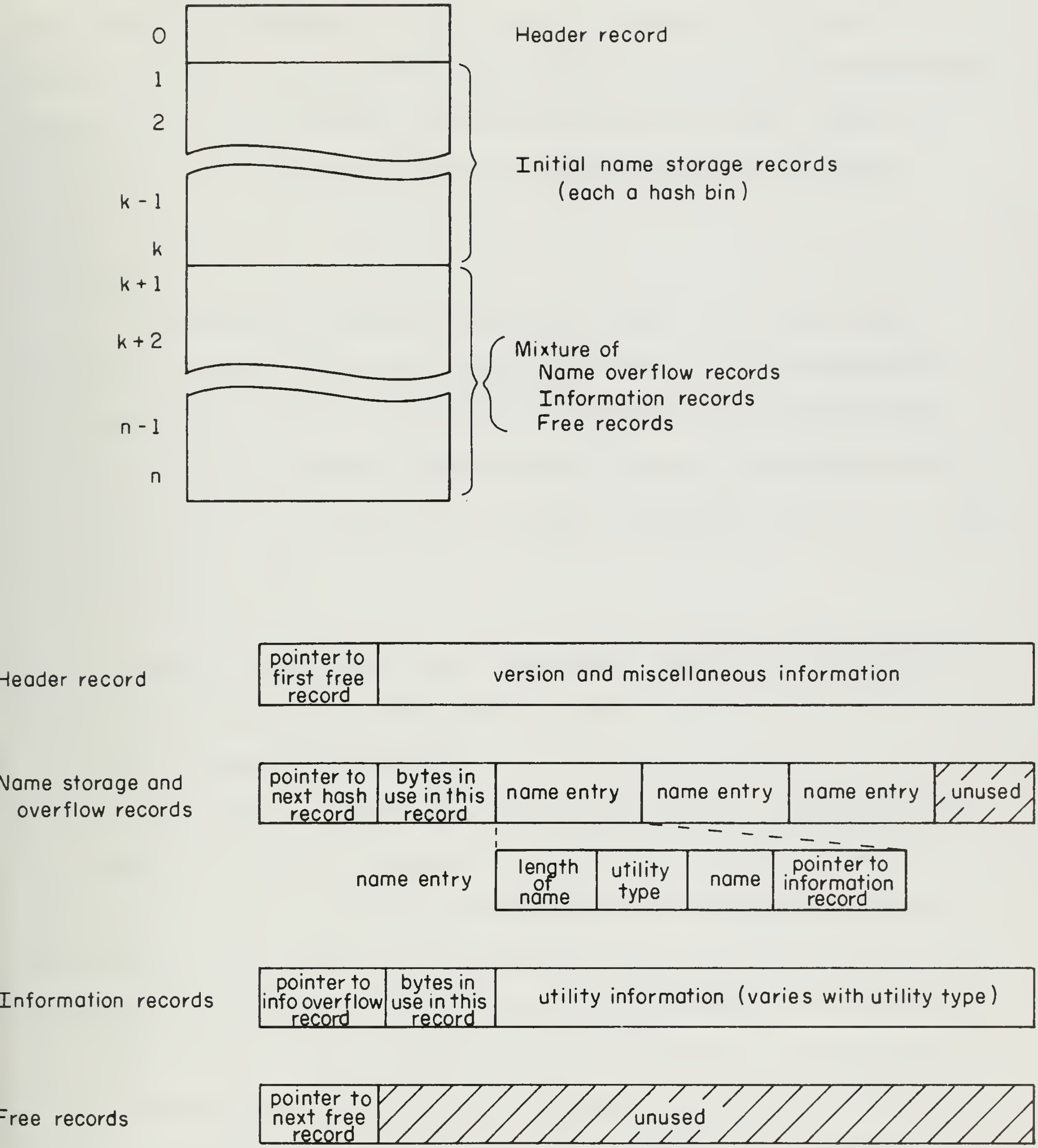


Figure 17: Utility File Structure

Generalized analysis packages can have access to IRIS/NARIS processed data through the output and calculate interface files. Some analysis packages create auxiliary files with IRIS/NARIS compatible geolocators in them. These can be accessed in the retrieval language and turned into IRIS/NARIS regions.

6. Critique

There were many things that were done right and at least as many things that were done wrong when building this system. We will describe only those points which had the greatest impact on IRIS/NARIS and those which are likely to be of most interest to the designer of any user oriented data system and, in particular, the designer of a geographic information system.

6.1 What Was Done Right

6.1.1 On Time Delivery

The production system was produced within the two year schedule and significantly exceeded the original specifications.

6.1.2 Need and Impact Anticipation

The system, as designed and implemented, correctly anticipated many user needs and the impact of interactive computing on the future demands of this class of user. Some of the analysis features were implemented without user knowledge. The users did not understand the need for them and would have insisted that we not work on them. If we had not implemented these features, it is clear that the system would not have been powerful and flexible enough to have attracted new users. It would not have met user needs that became obvious to users once the system was in use. If we had not over designed, and, instead, had implemented the system requested by the user, NARIS would have died at the end of the first two year grant.

6.1.3 Simple Data Structure

The simple data structure chosen has proved to be good. The data structure has been rich enough to accommodate all parcel attributes, and simple enough to allow a very efficient implementation. Even map coordinates and parcel specification data are stored in the parcel as normal data classes. This permits the parcel to be self defining and obviates the need for separate files to be used to generate tables and graphics.

6.1.4 User Acceptable Language

The part of the system most often praised by the users is the language. The language has been well engineered to this user class. Potential users often are able to use it for routine interrogations after a one hour introduction and demonstration.

6.1.5 Cost Effective System

The system is cheap to use. The planner normally operates with local dollars on a tight budget. IRIS/NARIS can be run at about a tenth of the cost associated with comparable systems investigated in the IRIS state-of-the-art survey [1].

6.1.6 Technology Push

At the time this system was designed, it was pushing at the edge of data management technology. The semi-automatic data compression, the default occurrence, and dynamic tuning of Boolean expressions during retrieval have all paid off and helped to make the system cost effective. The user defined, dynamic index (regions) allows the user to tune his requests and scope of action on a very large data base to his own needs. Users have been pleased with this ability. However, it may not be as useful on a smaller data base or one which changes more rapidly.

6.1.7 Code Documentation

From the beginning we insisted on professional, in-line documentation of the code. The documentation has been done well enough that we have survived the personnel changes that accompany a multiyear project and do not have an irreplaceable programmer problem.

6.1.8 Execution Time Traces and Measurements

Execution time trace and measurement facilities can be turned on for any selected set of the major procedures in the system. The traces produce detailed listings showing the flow of command and the parameters passed between procedures. The consumption of various system resources can be measured for each individual procedure or for various groups of procedures. The modest effort involved in producing these facilities was amply repaid in reduced debugging time and the acceleration of subsequent tuning efforts.

6.1.9 Extensive Data Cleaning

The data base inserter, preprocessor and symbol table were all designed to increase the redundancy of information at data input. Ample opportunity is provided and exploited for consistency checks and other cleaning activities. The cleaning system is so effective that it frequently discovers inconsistencies in machine readable data bases that were previously in use outside of the IRIS/NARIS system.

6.1.10 Missing Data Facilities

The system was designed from the ground up to recognize and correctly process missing data. Patchwork coverage of the geographic region is more common than complete coverage. Frequently, the data available from the data supplier contains missing data items for one reason or another. For example, geologic well logs have been recorded since the turn of the century. The older

logs did not record as much information as the more recent ones. However, the information contained in the older logs is still useful. The missing data facilities of IRIS/NARIS have permitted this data to be input and then analyzed in a meaningful way.

6.2 What We Did Wrong

In fairness to the design and programming staff who worked on the IRIS/NARIS project it should be noted that much of what we did wrong was imposed by the necessity of getting the project done on time in a limited budget. In particular, the need for items 2, 3, 5, 6, 7, and 9 below was perceived in varying degrees during the design phase. However, they had to be dropped from consideration due to time and manpower limitations. Our perceptions of the relative importance of each of these items is now more accurate. With hindsight, we would have modified few of our system priorities.

6.2.1 One User Dependence

We were particularly naive about the political situation when the project began. As a result, we depended too heavily on one user for input. By failing to consider the conflicting missions and attitudes of public agencies we got a distorted view of the user community that led to blunders 2 and 3 below.

6.2.2 Multiple Resolution is Awkward

Multiple resolution is central to the way planning data is captured and used. The original NARIS only had one data resolution. The current IRIS/NARIS can access several data bases of varying resolution. However, the use of data in two different data bases, for one analysis, is currently clumsy and, quite frankly, a kludge.

6.2.3 Clumsy Data Insertion

Data insertion is clumsy and time consuming. IRIS/NARIS has been built for production use of a large data base. A lot of attention is paid to cleaning the data to insure the highest quality data, and to protecting the data base from a castastrophic loss. There is a need for a quick insertion of data classes covering a small area. The planner needs a system that can accommodate the small and volatile data bases he needs in his fire fighting role. We see no technological reason why a data insertion system cannot handle the high quality static data base and the volatile data base with equal facility.

6.2.4 User Documentation

We didn't produce a primer for the user. To our knowledge, no user has read the tome we call the User Reference Manual. While the User Reference Manual is complete, it is simply too imposing to expect a user to read it. We do have shirt pocket references for the language, but something in between is needed. The closest thing to a primer is a document produced by a user agency to show other agencies how they use IRIS/NARIS [4]. A user is probably the best person to write the user manual. While awaiting the primer, we are putting references to the user reference manual into the on-line prompts and error messages. It is too early to know the results of this latter approach.

6.2.5 Scanner/Parser Weaknesses

The scanner/parser for the query language is not flexible enough. The current scanner, for example, does not have look-ahead. As a result, some of the phrase compilers gobble up too much or too little of the input string and cause correct syntax to be incorrectly parsed. Fortunately, this does not produce erroneous retrievals. Rather, a compiler error is flagged. The user can avoid the error by using parentheses.

6.2.6 No Distance Concept

There is no distance concept in IRIS/NARIS. Highly accurate map data is contained in the parcel for graphic output. This can and has been used to calculate distance, but it is clumsy. A reasonable amount of thought has been given to the problem of the syntax and semantics which should be used to implement a distance concept. We do not yet have a solution which fits the planning problem well.

6.2.7 No Very High Level Data Expressions

In the rush to meet schedule, the concept of very high level data expressions was dropped. This led to several verbs, notably *tabulate*, *calculate*, and *output*, which do similar things on slightly different kinds of expressions. The concept of a data expression should extend from the element continuously through to the region level. This would permit more powerful analyses and only one verb for table and file preparation -- hopefully a less confusing situation for the user. Most use of the *output* verb has been to circumvent some of the analytical weaknesses caused by this limitation.

6.2.8 Direct Indexing of Regions

It is fast to use a logical record address in the data base file as the internal parcel number. However, since hundreds of regions may contain that record number, it is very difficult to move parcel data around within the data base file. It would have been more flexible to store indirect addressing codes in regions. The system could use the parcel directory to translate the address codes into direct file references, at retrieval time, on the fly. This approach would also allow users to specify regions containing parcels which might later be added to the data base.

6.2.9 Table Facility for Data Interpretations

Much of the objectively measured "raw" resource data is difficult for non-experts to use. Plans for a utility (like abbreviations, functions, or regions) to allow storage of tabular transformations from broad data to useable intepretations was considered. Although the need for the facility was recognized and supported by the users, funds were not available for its implementation.

7. Conclusions

IRIS/NARIS, as a piece of technology, is one of the best geographic information systems available. Its user language has been well received, it is cost effective, it supports many of the analyses frequently needed by the planner, and permits easy access to external models and analysis packages.

IRIS/NARIS could have been done much better if more resources were available to implement features known to be desirable but infeasible to implement given the personnel and funding limitations. IRIS/NARIS is actually a rather modest system compared to what current technology should support. In fact, we were surprised to find, during the IRIS state-of-the-art survey, that the early NARIS facility compared so favorably with other geographic information systems. Most of the other systems were weak in one or more of the areas of costs, capabilities or human engineering.

IRIS/NARIS is just a beginning in the planning system area. Planners have special problems in the areas of reliability, costs, massive interactive data bases, rapid response, large searches per request, complex analyses and human engineering. New requirements for environmental impact statements, the existence of a more aware citizenry, and the tremendous pressures brought to bear on public decision makers will all continue to fuel the growth of these

systems. To meet the special requirements of these systems we will require sophisticated data management techniques and highly tuned retrieval algorithms and retrieval/analysis languages.

8. References

1. "IRIS Feasibility Study Final Report", Center for Advanced Computation, University of Illinois at Urbana-Champaign, April, 1972.*

2. Al Meyers, "MONICA User Manual", Center for Advanced Computation, University of Illinois at Urbana-Champaign.

3. Schuster, S. A., Locating Optimal Sites in Geographic Information Systems, Ph.D. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1973.

4. Hutzel, I. and al Chalabi, S., "IRIS Evaluation Experiments", Northeastern Illinois Planning Commission Research Memo #15, Chicago, Illinois, May 2, 1973.

---

* Includes an extensive bibliography and analysis of the state of the art in geographic information systems.