F I G XVIII. INTERNATIONAL CONGRESS

TORONTO, CANADA, 1986

LOBSTER

Combining Database Management and Artificial Intelligence Techniques to Manage Land Information

Eine Verbindung von Datenbank-Verwaltung-System und Methoden der kuenstlichen Intelligenz zur Verwaltung von Land Information

L'integration des methodes d'intelligence artificielle avec les systemes de banque des donnees pour l'administration des donnees a reference spatiale

Andrew U. Frank

Abstract

The overall design of Land Information Systems touches upon many different problems of organizational, cognitive and technical nature. We understand these three categories broadly and will shortly describe how we differentiate between them. The paper then concentrates on cognitive problems as encountered during our efforts to design the MAPQUERY language to retrieve geometric and alphanumeric data from a Land Information System. It is argued that the deductive power of logic inference systems, which are typical for Artificial Intelligence techniques, are necessary. Such methods are of considerable help to integrate data from different sources into a unified system. At the end we report on the implementation of LOBSTER, a system to integrate a Prolog like language with a network database.

Zusammenfassung

Der Entwurf von Landinformations-Systemen beruehrt manche organisatorische, kognitive und technische Probleme. Der Beitrag konzentriert sich auf die kognitiven Aspekte, die bei der Gestaltung der MAPQUERY Abfragesprache angetroffen wurden. MAPQUERY erlaubt dem Benuetzer geometrische und alphanumerische Daten aus einem LIS abzufragen und graphisch oder in Tabellenform darstellen zu lassen. Wir schliessen, dass eine Abfragesprache fuer ein LIS auf Methoden der kuenstlichen Intelligenz, insbesondere der automatischen Schlussmethoden und der deklarativen Programmierung aufbauen sollte. Diese Methoden sind besonders hilfreich, um semantische Differenzen zwischen Daten aus verschiedenen Quellen zu ueberbruecken und diese in ein einheitliches System zu integrieren. Zum Schluss berichten wir ueber die Implementierung von LOBSTER, ein System, das eine Prolog-artige Sprache in einer raeumlichen Datenbank integriert.

Resumee

La construction des systemes d'information du territoire peut etre considere sous aspect d'organisation, des problems cognitives ou techniques. Les contributions des sciences cognitives et les methodes de l'intelligence artificielle pour la realisation des SIT sont mis en relief. Ces methodes sont extremement valables pour la transformation des donnees codees differament dans un code uniform permettant leur integration. Dans nos efforts pour la realisation d'une language pour acceder des donnees stockees dans un SIT et leur representation graphique (MAPQUERY), nous avons trouve necessaire l'expressivite des languages basees sur logique et la possibilite des deductions logiques. Nous avons realise un system interactive utilisant une language de type Prolog, integree avec une banque des donnees spatiales.

Keywords

Artificial Intelligence, database query language, Prolog.

1 INTRODUCTION

In order to understand the overall design of Land Information Systems a theoretical framework is crucial. We need to understand which problems are related and which are not. Such a theoretical framework is important for the proper understanding of the user's needs, their categorization and for placing them into context. The same theoretical framework is also necessary for finding appropriate answers. The framework helps us to avoid building 'good' systems which just miss the user's demands (for example, if a customer complains about his car being not large enough do not waste your effort to sell him a faster one).

The theoretical framework for Land Information Systems is slow to come. Many have contributed from different aspects [Chevallier 1981] [Frank 1982a] [Frank 1983a] [Epstein 1984]. The task is larger than expected and requires contributions from different areas of science. It is clearly related to the area of information systems in general and to computer aided design and computer aided manufacturing (CAD/CAM) particularly. In this paper, methods from Artificial Intelligence research [Barr 1982] [Hayes 1983] and cognitive science are introduced to the LIS discussion [Peuquet 1983]. It will be shown how an LIS framework could be built on a three tier model of organizational, cognitive and technical aspects. Then we will discuss how typical expert system methods can be applied to LIS and how a flexible, 'intelligent' query language provides at least a better user interface, but also allows to address a number of other problems. Finally we will report how the LOBSTER system was built, integrating a Prolog like language [Clocksin 1981] with a network database [Frank 1982c] [Frank 1984a] [Frank 1984c].

2 A THREE TIER FRAMEWORK

Past discussion regarding a framework for an LIS often concentrated on very specific aspects, especially the technical design of the necessary computer systems [Frank 1981] [Frank 1982a]. Others have introduced a system oriented point of view [Chevallier 1981], with concentration on organizational aspects [Frank 1985a]. Indeed it has been argued that the LIS framework would consist of an organizational and a technical part [Chevallier 1980].

It seems, however, that it would be reasonable to separate a cognitive component and to differentiate it from the organizational and technical components. Without attempting a final definition, cognitive sciences deal generally with information and specifically with the meaning of words and concepts and how these affect formal treatment.

Computer systems are essentially formal systems for treatment of symbols; computers do not understand - in the sense of human beings' understanding - what the symbols mean, but rather manipulate them according to rules laid down in a program. The behavior of computer systems can be described by formal theories based on mathematical logic. The apparent processing of information in a computer system is nothing more than a simulation of the behaviour of human beings using formal rules. Writing programs is to formalize the rules and methods a human being uses in a formal language (eg. calculating the number of days between two dates).

In the concept of a Land Information System this means that we separate the following three aspects:

2.1 Organizational Aspects

Information, and especially information about land, can lend power to those who have access to it. Society has built checks and balances to achieve equitable solutions and rules of 'fair process' to decide about issues of distribution of power between organizations. These primarily legal rules contain methods for their extension to new situations, but the legal system in most countries has not yet found final solutions for the problems introduced by the new information technology. The design of Land Information Systems needs equitable distribution of the gains of such systems and rules to safeguard against unilateral exploitation.

In many cases, Land Information Systems are used by only one organization, eg. a utility company or the public administration. Then conflicts of interest between groups (organizational units) within the parent organization, which are similar to conflicts between different organizations, can appear and need solutions, without the benefit of a legal framework. Studies of organizations and how they react to information systems have been done (especially in the management information system area), but their results are not very conclusive.

Further, the information system and reality interact in many ways. Studies of such interactions and descriptions of their rules are best done in the organizational context, using system theory approaches [Chevallier 1981].

2.2 Technical Aspects

Most obvious, but not most important, the technical aspects of Land Information Systems deal with the selection of hardware (computer system) for the processing of the data.

For LIS which do not use electronic data processing equipment, technical concerns are the selection of cartographic materials and procedures, etc. Often technical discussions concentrate on surveying methods and scale of base maps for LIS and their dependency on the purpose of the LIS, the need for permanent monuments on the ground etc.

We have argued in the past that the production and the design of the programs used for a LIS are equally important technical problems, and stressed advanced software engineering methods [Guttag 1977] [Parnas 1972]. The implementation of the LOBSTER system applied these methods in the overall concept as well as in the details of the programming. It demonstrates that the multi layer model is appropriate as a (software design) framework for the design of a LIS. LOBSTER can be seen as a query language and a human interface to the LIS, and it did easily fit within the existing software.

2.3 Cognitive Aspects

The occupation with logic based programming [Kowalski 1979] and Artificial Intelligence methods in general has shed some new light on certain typical and widely discussed LIS problems. The technical realization of an LIS using a computer system uses a formal description of its data processing methods in programs. Independent of the specifics of these programs (eg. the programming language used, the software engineering methods, the hardware etc.), the automatic 'information' processing can be seen as a formal system (a mathematical theory), which might be described in eg. a first order logic. On this level we have abstracted from all material problems, such as amount of storage used, performance or price of the system, and concentrated on the logic of the symbol manipulation process - or, as seen from a users point of view, on the information transformation.

If the information system is to work properly, the formal theory must reflect the understanding human beings have about the meaning of the symbols and how they are related to each other. This happens at different levels, eg. at a relatively simple level based on "naive" mathematics in order to calculate the surface area of a given parcel, or at relatively complex level of legal reasoning about the order of precedence and the amount of mortgages on a property.

Chevallier reported the typical problem of different meanings of a term, eg. 'road width' for the surveyor (width of the right of way), for the traffic engineer (width of the circulation channel) or for the contracter (width of the hard covered surface). Similar differences can be found in most instances where data from different sources need to be integrated. The conceptual level is most appropriate for dealing with this type of a problem. We argue that these are not organizational problems which can be overcome by enforcing uniform standards for encoding across different departments, because we assume that the specialists have reasons for their use of terms. Pressure to force them to use standardized terminology will more likely cause resistance against the use of the system. However, if the terms for each speciality can be properly defined, formal methods to translate between them should be available.

Guttenberg has drawn our attention to the differences in names used for things depending on the interest of the observer and found that classification of use of land or buildings is difficult. Not only limits between different uses often are difficult to draw (when does the use of a building switch from multi family dwelling to an office? 50 % of the apartments or 50 % of the surface ?), but other, seemingly trivial classifications may cause problems: consider a typical church building, but used now as a public library - should it be classified as a church or as an office building?[Guttenberg 1979] [Guttenberg 1981].

If the stored data are used to deduce other quantities from them, differences in their original meaning must be considered. If for example a classification schema is only very loosely defined, the aggregated data may not be used to deduce changes over time - the observed changes may as well be changes in the way the classification is applied.

In most general terms, cognitive aspects of LIS include all questions of how we describe reality and how this original information is transformed in system responses to user queries. This includes the operators we provide to users to formulate their queries as well as the conceptual tools used to describe the information transformation within the system.

Cognitive problems are more important for LIS than for other information systems, because LIS are designed for multiple uses by different organizations. Integration of information from different sources is difficult. The major problems are the differences in meaning and concepts used by the different organizations which originally collected the data. We critically need tools to describe differences in the semantics or aspects of data quality in data collections in order to be able to predict the interaction of data from different sources.

Mathematical logic can form the fundament for such methods and 'programming in logic' [Kowalski 1979] seems to be a useful tool. The following description of logic-based programming and its use for LIS should give the reader a more practical grasp what can be achieved today. We describe programs which currently work, but the techniques are still quite experimental and need further research work before they can be introduced in the practical surveyor's office.

3 THE PROLOG LANGUAGE

The Prolog language is based on mathematical logic (precisely first order logic) and, unlike other programming languages, does expect statements about what is true (not what to do). Thus a program written in Prolog reads like a collection of true statements, each of them is easy to verify. The Prolog interpreter then automatically makes the connections between the statements and establishes the truth of other statements. This is called 'declarative programming' (you declare what is correct) and contrasts with the traditional 'procedural programming' style, where the programmer commands which steps to do in what order.

Such automatic deduction of conclusions from basic facts can only work efficiently if we accept certain limitations in the form the facts can be presented in. Prolog uses so called 'Horn' form, where a fact is expressed as predicate with some arguments, eg.

```
father (Adam, Beat).
```

stating that 'Adam is the father of Beat',

or more complex rules, where the truth of a predicate is dependent on the truth of some others:

```
grandfather (g, s) if father (g,p) and father (p,s).
```

stating that 'g is grandfather of s, if g is father of p and p is father of s' (note: constant symbols start with upper case letters like Adam, Beat, and variables with lower case letters, eg. g, p, s).

To complete our example, we add a few more facts:

```
father (Beat, Caesar).
father (Caesar, Daniel).
```

and we can now ask the Prolog interpreter:

```
father (x, Caesar).
```

and receive the answer ' father (Beat, Caesar)' informing us that according to the given data Beat is the father of Caesar. We can also ask who is the son of Caesar:

father (Caesar, x).

and receive the answer ' father (Caesar, Daniel)' informing us that according to the given data, Daniel is the son of Caesar. For these two queries, only a search of the data base was necessary.

Prolog, however, permits more complex questions, like, who is the grandfather of Caesar:

grandfather (x, Caesar).

and it will use the above stated rule to find 'Adam'. It works not only in such simple cases, but Prolog generally permits to state facts in form of predicates or as rules, where a fact depends on the truth of other predicates (which in turn may depend on other rules).

Prolog interpreters can be found for many computers, from personal computer to mainframe. Several good texts describe the full power of Prolog [Clocksin 1981] [Clark 1984], and anybody interested in computer programming and LIS is well advised to read them.

Prolog can not only be used for retrieving data from a small database but has the full power of a general programming language (eg. Pascal or FORTRAN). It was originally built to help with the understanding of natural language text, but has since been used in many different projects. The major disadvantage of Prolog was its relative slow execution speed. Most Prolog implementation are interpreters, but fast compilers become available now and some claim that Prolog programs can execute as fast as a program written in a procedural language.

4 LOBSTER - A SPECIAL PROLOG INTERPRETER

Users need tools to retrieve data from a database. The previous section should have convinced that Prolog is well suited for data retrieval. However, most Prolog implementations are not prepared to store the large number of facts that a LIS must contain (eg. the millions of point coordinates etc). Thus we concluded that a Prolog like language could be used as a user interface or query language to a regular LIS, based on a database suited for dealing with large numbers of spatial data records. The database for a LIS must respond to a number of requirements[Frank 1984b], eg.

- object oriented database design,

- generalization/spezialization available for data structuring,
- suitable for modelling geometric data,
- fast access based on spatial location, etc.

We had spent considerable effort to build such a database [Frank 1982c] [Frank 1983a] [Frank 1983b]. Thus we decided to use this database for storing the LIS data and to use the Prolog interpreter and language for building a query language. It was necessary to build a path for the Prolog interpreter to access the data stored in the LIS, and to have stored data appear as Prolog facts. This was achieved in two steps: First, coding procedures for database access and integrating these with the Prolog interpreter such that they appear as regular Prolog predicates (so called built-ins); this allows a low level of access to database facts from Prolog. Second, we had to define mappings from the conceptual database schema to Prolog predicates, which defines how database facts appear under Prolog, and implement these mappings as Prolog rules using the database access built-ins.

We constructed other Prolog built-in predicates for graphical output and can now use Prolog to retrieve data from the database and to show them graphically. So far we found the Prolog language to be a fast and easy way to formalize the complex rules for query processing and output generation in LIS query languages.

At the time this work started, not many versatile Prolog interpreters were available and we decided to write our own code. LOBSTER is slightly different from a standard Prolog in the following aspects:

1. permanence of rules and facts: rules and facts a user stores are kept in a permanent database and are available for any future work (standard Prolog demands that the used rules and facts are loaded at the beginning of each session).
2. organization of rules and facts into groups: making rules and facts permanent required some organizational tools for the user to keep track of what he had previously defined.
3. extendability: new built-in predicates can be added easily.

In the meantime a number of other research groups have approached similar issues and some of the Prolog interpreters now commercially available would at least partially fulfill our requirements (especially the possibility to integrate code written in standard high level programming languages).

5 QUERY LANGUAGES FOR LIS

The query language of a database system is the language a user uses to express to the database what data he needs and how it should be presented. Despite considerable efforts the ultimate query language for standard commercial databases has not been found. Presently the SQL language used in IBM's relational databases seems to emerge as a standard for commercial applications; nevertheless, it is not very easy or natural to use and is limited in its expressive power (no loops or recursion). Many new proposals are presented at every database conference, which indicates that the ultimate solution is not yet found.

Few proposals have been made for a database query language to deal with graphics output [Frank 1982b]. Our previous implementation of MAPQUERY [Egenhofer 1984] was based on SQL and suffered from the lack of a general loop construct or recursion. This becomes most apparent for a request like:

Find a specific forest parcel and return the largest connected forest area this parcel is part of.

This is logically translated to:

1. find the parcel x and place it in a set s.
2. Repeat till the set does not grow any more:
   for each parcel p in s find all its adjoiners and if the parcel type is forest, add them to the set s.

Technically we call this transitive closure and it is a typical example for a least fixed point operation. It can be implemented using a loop or a recursion, but both of these devices are missing in SQL and therefore it is not possible to express these (and all similar) queries in SQL. Prolog like languages include recursions and thus have no problems to express the above query.

Further, a query language to be used in a multi purpose LIS must provide means to define the terms used for one purpose in terms of another one. This is extremely simple in Prolog (or LOBSTER) and can even be done by a user. For example, a user wishes to define a motorway as a road that is used by automobiles and has a width of at least 8 m:

```
motorway (x) if usedBy (x, automobiles) and width (x, w) and
larger (w, 8).
```

From then on, the user can use the term motorway with this specific definition. In LOBSTER we plan to include facilities so that each user (or user group) can include his own definitions. These definitions then can be used to define others, for example personal abbreviations.

For cartographic query languages it is necessary to complete a user query (which may ask for a specific point) with a statements to retrieve a suitable environment (context) for presentation. This completion must be done considering the user and the task he tries to accomplish. In other Artificial Intelligence projects, attempts to accumulate information about the user and his interests have been made; similar efforts are necessary for LIS and it seems that a Prolog like language can easily support such efforts.

6 FORMAL LOGIC FOR DEFINITION OF TERMS

Discussion of terminology between application domains is primarily hindered by the difficulty to percisely explain what is meant by a term (the semantics).

The selection of different terms for the same thing or using the same term for different things do not cause problems, as long as the use of the terms is consistent for each user (group) and exact definitions exist. It is then possible to translate the terms into some artificial standard terms, using Prolog rules [Frank 1985b] [Frank 1985c].

The correct and unambiguous definition of terms, however, is much more difficult and subtle differences can have considerable effects. Prolog rules can not only be used to retrieve data from a database, but they can also be interpreted as formal definitions for terms. Palmer reports the use of Prolog rules to retrieve hills and valleys from a irregular triangulation terrain model[Palmer 1984]. The same rules can be read as formal definitions for the terms 'hill' and 'valley'. These definitions are precise and do not leave room for differing interpretations (they are based on other, formally defined terms and are ultimately based on a triangulation of the surface and the height of points). Their intuitive correctness can be observed when used to retrieve data, but they can also be examined and formal deductions made from them.

7 CONCLUSIONS

The methods and tools Artificial Intelligence research has developed over the past decade are very helpful for the construction of intelligent Land Information Systems. A logic based language like LOBSTER provides the additional expressive power standard which commercially oriented database query languages do not have, and which is necessary for geometry oriented data. The interface with the database can be selected such that the resulting language is not more complicated than standard SQL.

Moreover, a logic oriented language allows experts to set up the language such that the differences in terminology between user groups of a LIS can be accomodated. It seems also possible to build systems that note the user's behavior over time, extrapolate and meet his expectations automatically.

Finally, a logic oriented language draws our attention to formal methods to specify our terms and thus helps to overcome the misunderstandings in natural language terms. Formal definitions do not only help programmers, but generally improve our understanding of things.

Granted, all this can be achieved using the formal logic and any programming language. The difference between a system like LOBSTER and other traditional solutions is the separation of concern between the logical inference system, and the user definitions. Using procedural languages these aspects are tightly interwoven, and it is therefore more difficult and more expensive to write the programs and very time consuming to change them. The definition in a Prolog needs not to be written by a programmer, but can be done by an expert in the application domain.

BIBLIOGRAPHY

Barr, A., Feigenbaum, E.A., The Handbook of Aritificial Intelligence, Pitman, London 1982

Chevallier, J.J., Land Information Systems - a global and system theoretique approach, paper 301.2, Federation Internatinale des Geometres, XVI Congress, Montreux (Switzerland), 1981

Chevallier, J.J., De la mensuration cadastrale a un systeme d'information du territoire, revolution ou evolution, Surveying, Photogrammetry and Rural Engineering (VPK), no. 2, Feb. 1980, Zurich (Switzerland)

Clark, K.L., McCabe, F.G., micro-Prolog: Programming in Logic, Prentice Hall international SEries in Computer Science, 1984

Clocksin, W.F., Mellish, C.S., Programming in Prolog, Springer Verlag, New York, 1981

Egenhofer, M., MAPQUERY - Query Language for Land Information Systems, Report Nr. 79, Institute for Geodesy and Photogrammetry, Swiss Federal Institute of Technology, Zurich (Switzerland), 1984

Epstein, Earl F., Duchesneau Thomas D., The User and Value of a Geodetic Reference System, Federal Geodetic Control Committee, 1984

Frank, A., Land Information Systems - theoretical and practical problems, paper 305.1, Federation Internatinale des Geometres, XVI Congress, Montreux (Switzerland), 1981

Frank, A., Conceptual Framework for Land Information Systems - A First Approach, paper presented to the 1982 Meeting of Commision 3 of the FIG, Report No. 38, Surveying Program, Universtity of Maine at Orono, 1982

Frank, A., MAPQUERY, Data base query language for retrieval of geometric data and its graphical representation, Proceedings SIGGRAPH '82, Computer Graphics, Vol. 16, No. 3, July 1982, p. 199

Frank, A., PANDA: a Pascal network database management system, in: W. Gorsline, Ed., Proceedings of the Fifth Symposium on Small Systems, ACM SIGSMALL, Colorado Springs, Col., 1982

Frank, A., Data Structures for Land Information Systems - Semantical, Topological and Spatial Relations in Data of Geo-Sciences, Ph.D. Thesis, Swiss Federal Institute of Technology, Zurich (Switzerland), 1983

Frank, A., Storage methods for space related data: the FIELD TREE, in: M. Barr (Ed.), Proceedings Spatial Algorithms for Processing Land Data, Symposium of the Lincoln Institute, Cambridge, Mass., 1983

Frank, A., Extending a network database with prolog, in: L. Kerschberg (Ed.), Proceedings First International Workshop on Expert Database Systems, Kiawah Islands, S.C., 1984

Frank, A., Requirements for database systems suitable to manage large spatial databases, in: D. Marble, et al. (Eds), Proceedings of the International Symposium on Spatial Data Handling, Zurich (Switzerland), 1984

Frank, A., Combining a Network Database with Logic Programming, Report No. 45, Surveying Program, Universtity of Maine at Orono, 1984

Frank, A., Distributed Databases for Surveying, Journal of Surveying Engineering, American Societey of Civil Engineers, Vol. 111, No. 1, March 1985

Frank, A., Computer Assisted Cartography, Lecture Notes, Surveying Engineering Program, University of Maine at Orono, 1985

Frank, Integrating Mechanism for Storage and Retrieval of Data, in: Onsrud, H.J., Workshop on Fundamental Research Needs in Surveying, Mapping, and Land Information Systems, Report, Virginia Polytechnic Institute and State University, Blacksburg, Va, 1985

Guttag, J., Abstract data types and the development of data structures, Comm. of the ACM, June 1977

Guttag, J., et al., The design of data specifications, , in: Yeh, R.T. (Ed.), Current Trends in Programming Methodology, Vol. 4, Data Structuring, Prentice-Hall, 1978

Guttenberg, A., Land data classification - a linguistic approach, Proceedings of the 7th European Symposium on Urban Data Management, The Hague, 1979, paper 8

Guttenberg, A., Uniformity and flexibility in the classification of topographic data, paper 301.4, Federation Internatinale des Geometres, XVI Congress, Montreux (Switzerland), 1981

Hayes-Roth, F., Waterman, D.A., Lenat, D.B., Building Expert System, Addison Wesleyu, Reading (Mass.), 1983

Kowalski, R., Logic for Problem Solving, Elsevier, New York, 1979

Palmer, B. Symbolic feature analysis and expert systems, in: D. Marble, et al. (Eds), Proceedings of the International Symposium on Spatial Data Handling, Zurich (Switzerland), 1984

Parnas, D.L., A technique for software module specification with examples, Comm. of the ACM, Vol. 15, No. 5, May 1972

Peuquet, D., The application of artificial intelligence techniques to very large geographic databases, Proc. Sixth Internat. Symposium on Automated Cartography, Ottawa, 1983, Vol. 1,pp. 419-420.

Address of author:

Dr. Andrew U. Frank
Surveying Engineering Program
Boardman Hall 103
University of Maine
Orono, Maine 04469
USA